Research on SSL/TLS Security Differences Based on RFC Documents

1st Ziqiu Zheng
Department of Guangzhou Institute of Technology
Xidian University
Guangzhou, China
zhengzq@nipc.org.cn

2nd Xuejun Li
Xidian University
Xian, China
aluckydd@mail.xidian.edu.cn

3rd He Wang
Xidian University
Xian, China
hewang@xidian.edu.cn

4th Gaofei Wu
Xidian University
Xian, China
gfwu@xidian.edu.cn

5th *Yuqing Zhang
Xidian University
University of Chinese Academy of Sciences
Hainan University
zhangyq@nipc.org.cn

***Abstract*—The RFC standard document provides a detailed description of how a system is designed and correctly used to maintain security. The document itself may already contain information for predicting the existence of certain security issues. RFCdiff is a tool that uses NLP technology to automatically extract rules from RFC documents, identify differences and assess their compliance with RFC specifications. Through evaluations of well-known TLS implementation libraries such as OpenSSL, mbedTLS, GnuTLS, and WolfSSL, total of 56 security issues due to differences were identified. This research offers a valuable reference for potential issues in the implementation of SSL/TLS.**

***Index Terms*—RFC, SSL/TLS, Difference testing.**

I. Introduction

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), stand as widely adopted security protocols on the Internet, ensuring data confidentiality and integrity. The TLS protocol finds extensive application in various domains such as web browsing, email, instant messaging, transaction payments and more. This protocol is not only the security basis of the HTTPS [1] protocol, but also the basis of Internet security communication.

This work was supported by the National Key Research and Development Program(2023YFB3106400, 2023QY1202), the National Natural Science Foundation of China (U1836210), and the Key Research and Development Science and Technology of Hainan Province (GHYF2022010).

TLS is crafted on the foundation of open standards. The RFC document issued by the Internet Engineering Task Force (IETF) serves as the definitive source, providing specifications and guidance for the TLS protocol. This enables various vendors and organizations to implement and support the protocol. The IETF consistently releases new RFC documents to enhance and update the TLS protocol, encompass aspects like the encryption algorithm and security extensions.

The RFC standard document meticulously delineates the system's design and proper usage for maintaining security. Documentation frequently contains explicit or implicit indications of potential security risks, including warnings about destructive operations conflicting with security requirements or clues implying implementation errors, such as the absence of descriptions regarding crucial security checks.The document itself may already encompass information for anticipating the presence of certain security issues, it provides a method for discovering security issues within protocols.

II. Related work

In recent years, some work has tried to use text analysis technology to assist in the discovery of various vulnerabilities in the security field. Chen et al. [2] demonstrated that system documents contain a significant number of vulnerability-related

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

indicators. Through automated checks against the security requirements of payment models and services, they discovered 5 vulnerabilities that could lead to payment evasion. Kakarla et al. [3] detected RFC compliance errors in DNS name server implementations by employing automated test generation. Their assessment of 8 open-source DNS implementations resulted in the identification and reporting of 30 new, unique errors. Shen et al. [4] introduced the document analyzer Hdiff to extract rules from RFC specifications of HTTP and utilized differential testing to uncover semantic gap attacks. This effort revealed 3 such attacks across 10 popular HTTP implementations, identifying a total of 14 vulnerabilities and 29 affected server pairs. Kim et al. [5] conducted a syntactic and semantic comparison of extracted message structures with those specified, revealing 9 error cases, including 5 functional errors and 4 memory-related errors that violated implementation compliance with the specification; Wang et al. [6] found deviations from RFC specifications in email services, illustrating that attackers could leverage these differences to bypass security mechanisms and present deceptive results to email clients. It can beseen from the above related works that documents have been proven to be used to discover security issues.

A. Challenge

RFC specifications are described in natural language, which is an unstructured language and usually contains multiple layers of implicit semantics. Extracting rules from informal language descriptions such as RFC specifications is difficult. RFCdiff designed a document analyzer module, which automatically extracts natural language specification constraints and ASN.1 structure protocol rules from RFC documents based on NLP technology.

Sentences in RFCs are often lengthy and complex, and there are likely to be multiple parallel clauses. RFCdiff analyzes the dependency tree of complex sentences, and then identifies the parallel structure in the sentence, so as to divide a complex sentence into multiple simple clauses, and analyze the textual implications of each simple clause separately to obtain the target sentence full semantics more accurately.

Coreference Resolution. There may be some phrases in RFC documents that refer to relationships across sentences, such as pronouns such as ”it”, definite papers such as ”the”, quantifiers such as ”one”, and sequence values. This paper implements a forward search algorithm based on keyword fuzzy matching, trying to search for referred semantic phrases within three sentence distances , after finding the referred sentence, combine the two sentences into a complex sentence and then analyze the textual entailment relationship.

The majority of semantic difference problems entail logical safety issues, which typically do not exhibit clear error signs, such as crashes or memory corruption errors, making accurate detection challenging.Differences between specifications and implementations will be marked as potential errors. RFCdiff can locate the source of the vulnerability and further determine whether the difference will lead to security issues.

III. RFCDIFF DESIGN

During the initialization process, the form of the ASN.1 structure, the keyword dictionary, and the form of natural language expression are defined. After the initialization is completed, the traditional string regular matching method for obtaining rules is improved, and NLP technology is proposed to perform sentence segmentation, parsing dependency trees, text implication, text annotation, and pronoun resolution on RFC documents, which improves the accuracy of obtaining rules. Statements corresponding to keywords are extracted from the RFC for analysis and experimentation, with the addition of paragraph position markers, it can be further determined whether the difference conforms to the RFC specification, and the root cause of the discovered vulnerability can be quickly located.

A. Document rule extraction module

RFC rule extraction mainly uses the technology of automatic keyword retrieval and NLP technology. Through the automatic retrieval process, complex and irregular natural language is generated into sentences that can be recognized by computers. The sentence generation rule set is processed by NLP technology to obtain a rule data set that can be recognized by the computer.

In the RFC for TLS, rules are expressed in two ways. Specifically, most rules are written in Natural Language (NL), while some rules (in RFC 5280) are written in Abstract Syntax Notation (ASN.1).

RFC protocol specifications usually contain a series of sentences to explain the specific requirements of the specification for protocol implementation. RFC (Request for Comments) 2119 stipulates that rules must use the modal keyword to indicate the level of requirement. Modal keywords include:

(1) absolute requirements or prohibitions of specifications: MUST,REQUIRED, SHALL, MUST NOT, and SHALL NOT;

(2) requirements or prohibitions with flexibility: SHOULD, REC-OMMENDED, SHOULD NOT, and NOT RECOMMENDED;

(3) truly optional items: MAY and OPTIONAL. RFC 8174 emphasizes the use of uppercase keywords to facilitate the extraction of rules from RFC.

Specifically, the data in the paper is formatted and preprocessed first. The original RFC generally includes header, footer, body, title data, etc, the regular expression finds the line where the relevant header and footer are located, removes the header and footer, keeps only the sentence, and breaks the obtained plain text to ” .” or ASN.1 to split it into sentences that can be recognized by NLP techniques. Search for paragraphs containing keywords, record the current sentence and related chapter information, and save them into the natural language rule set. A large number of invalid characters are included in the sentence segmentation process, such as spaces and carriage returns, etc. All the words in the line are aggregated into a word set for word segmentation, and the form of the ASN.1 structure, the keyword dictionary, and the form of natural language expression are defined , to extract information from the statement. RFCdiff uses dependency tree analysis and text entailment techniques to identify the semantics of sentences, and to infer the logical semantic relationship of protocol specifications. RFCdiff identifies the key information in the sentence through dependency tree analysis technology, and fills it into the protocol specification template to obtain the protocol specification instance. Finally, RFCdiff infers the textual implications of sentences, and classifies sentences described in natural language into protocol specification seed assumptions. The design flow chart of RFCdiff is shown in Figure 3.1.

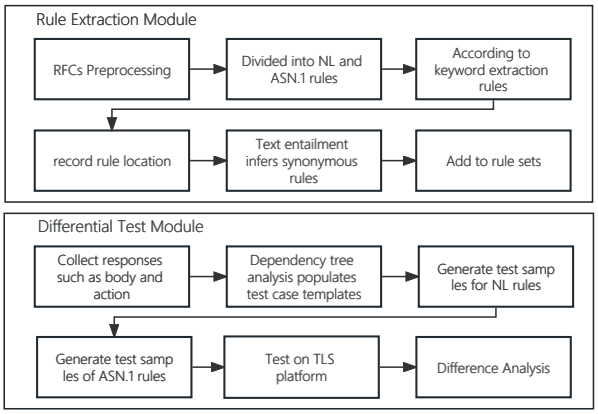

Fig. 3.1 Design flowchart of RFCdiff

B. Document Difference Analysis Module

Protocol specifications extracted in the document analysis module are translated into test cases with assertions. If the protocol implementation violates the assertion during the testing phase, the target implementation is considered to violate the protocol RFC specification. RFCdiff defines a series of status, such as valid, invalid, too long and role action phrases such as close connection, report error etc. The former is used to automatically generate test cases, and the latter is used for subsequent differential analysis. RFCdiff generates a series of rule-compliant test cases, which are then tested with mutated test data.

For the difference analysis component, RFCdiff will use the same test case to test multiple SSL/TLS implementation libraries, and find semantic differences by comparing the processing behavior of the SSL/TLS implementation libraries. Specific processing behaviors can be described using certificate verification, warning feedback, debugging logs, etc. In order to analyze and evaluate the test results conveniently, this paper defines: $\overrightarrow{DiffMetrics} =< id, degree, cond\_bd, cond\_val, assert(res\_bd, res\_val) >$.

Among them, id represents the unique ID of each test, degree is the degree of compliance (such as MUST, etc.), cond_bd is the condition entity, cond_val is the entity variable, assert is the assertion, res_bd is the result entity, and res_val is the result variable. Under different test requirements,

TABLE 4.1 Experiment data set

RFC	Brief description	Rules
5246 [7]	The Transport Layer Security (TLS) Protocol Version 1.2	124
5280 [8]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	158
7366 [9]	Specifies an Encrypt-then-MAC extension in response to CBC padding oracle attacks	5
7685 [10]	A Transport Layer Security (TLS) ClientHello Padding Extension	2
8422 [11]	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier	34
8446 [12]	The Transport Layer Security (TLS) Protocol Version 1.3	256

different $\overrightarrow{DiffMetrics}$ detection rules can be defined to discover semantic differences. Traditional difference testing techniques cannot locate the cause of the difference, nor can it know which implementation has a problem. In contrast, RFCdiff can determine whether a difference complies with the RFC specification due to the extraction of RFC rule information, and quickly locate the root cause of the difference. But RFCdiff can test individual implementations by checking whether $\overrightarrow{DiffMetrics}$ matches assertions from the rules.

IV. EXPERIMENTS AND FINDINGS

A. Configuration

The experiment was tested on 4 popular SSL/TLS implementations, and the following independent versions of the implementation library were tested differently: OpenSSL 1.1.1, mbedSSL 2.25.0, GnuTLS 3.7.0 and Wolfssl 5.5.0 .Experiments are performed on a virtual machine: Ubuntu x64 v18.04-LTS configured with i7-4790 CPU (3.60GHz) CPU core and 8 GB memory.

The data sets are RFC 5246, RFC 5280, RFC 7366, RFC 8422, RFC 8446, including 125,201 words and 2,116 valid sentences, a total of 492 NL rules and 87 ASN.1 grammar rules were extracted, and 358 test cases with assertions, and 2,627 test data are generated based on the test case generator. As shown in Table 4.1.

In the direction of SSL/TLS difference testing, this paper will compare with previous work. Frankencert [13] collected 243,246 certificates online, and reassembled the certificate components to generate 8,127,600 new certificates, and conducted difference tests on 15 implementations of SSL/TLS, and found 9 different differences. Blindness leads to low efficiency in finding differences; guided by code coverage, Mucert [14] used MCMC to mutate 1005 seed certificates, and found 27 different differences in 9 implementations; Symcerts [15] constructed symbols and The certificate chain with mixed specific values combined with optimization in specific fields, for SSL/TLS, 48 inconsistencies were found in 9 versions of 4 implementations.RFCdiff found a total of 56 inconsistencies on 4 implementations, and 21 inconsistencies were found on Openssl; 8 inconsistencies were found on mbedSSL; 21 inconsistencies were found on GnuTLS; and 3 inconsistencies were found on Wolfssl. The comparison is shown in Figure 4.1. Experiment result shown that the average number of differences found by the RFCdiff proposed in this paper is 14 in each SSL/TLS implementation, which is higher than 1 of the Frankencert , 3 of the Mucert , and 6 of the Symcerts.

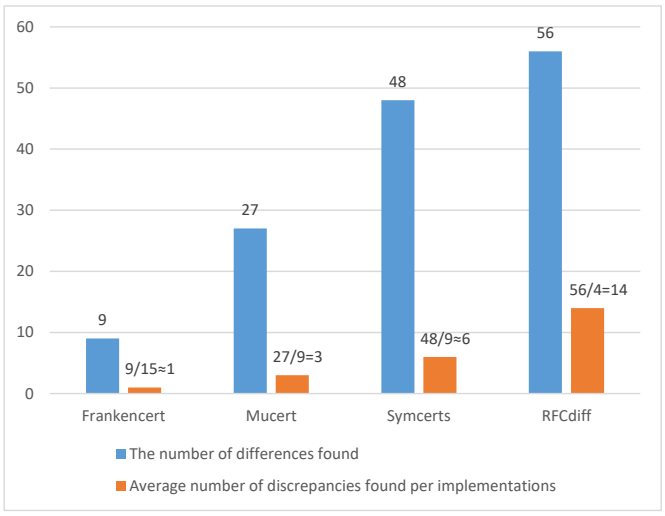

Fig. 4.1 The number of differences found by SSL/TLS implementations

B. Examples of Differential Security Questions

In openssl 1.1.1, RFC 5280 4.2.1.4 states that "If this extension is critical, the path validation software MUST be able to interpret this extension (including the optional qualifier), or MUST reject the certificate." The experiment found that: the test construct certificate "certificate When the extension of "policies" is "critical" (critical indicates whether it is a critical extension), openssl mistakenly accepts the certificate and successfully passes the certificate chain verification, instead of "MUST reject the certificate" specified by RFC. Malicious CAs can exploit this issue to deliberately assert invalid certificate policies in order to completely circumvent policy checks on certificates.

In addition, in gnutls3.7.0, RFC5246 states that "The end-entity certificate provided by the client MUST contain a key that is compatible with certificate_types; This message is only sent if the server requests a certificate. If no suitable certificate is available, the client MUST send a certificate message containing no certificates." The experiment found that: DSS_SIGN requires a DSA key, and the client certificate contains an RSA key, expecting that the client should respond with an empty certificate, but the client generated a non-empty Certificate message whose public key is not compatible with the certificate_types in the Certificate Request. The server may abort the handshake when receiving the wrong type of certificate from the client, and avoid the handshake if the client sends an empty certificate message. This problem can cause both parties to fail to complete the handshake.

V. CONCLUSION

The work of this paper has certain transferability and limitations. In terms of portability, the document has been proven to be able to detect security issues and has been used in payment services [2], DNS [3] and HTTP [4]. In the future, it can be used in other directions, such as the IoT protocol MQTT. In terms of limitations, if the RFC document does not provide a standardized description of the correct use of operations that may cause security problems, it will result in the inability to discover all security vulnerabilities. In addition, in order to extend RFCdiff to conduct new tests, need to manually create Test templates for further manual analysis. Although test template generation requires a lot of work, this work only needs to be performed once, and many test cases can be automatically generated using parameterizable test templates. The resulting test templates can be used to test different libraries in different versions, so this work is still meaningful.

REFERENCES

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," Tech. Rep., 1999.

[2] Y. Chen, L. Xing, Y. Qin, X. Liao, X. Wang, K. Chen, and W. Zou, "Devils in the guidance: predicting logic vulnerabilities in payment syndication services through automated documentation analysis," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 747–764.

[3] S. K. R. Kakarla, R. Beckett, T. Millstein, and G. Varghese, "{SCALE}: Automatically finding {RFC} compliance bugs in {DNS} nameservers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 307–323.

[4] K. Shen, J. Lu, Y. Yang, J. Chen, M. Zhang, H. Duan, J. Zhang, and X. Zheng, "Hdiff: A semi-automatic framework for discovering semantic gap attack in http implementations," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2022, pp. 1–13.

[5] E. Kim, D. Kim, C. Park, I. Yun, and Y. Kim, "Basespec: Comparative analysis of baseband software and cellular specifications for l3 protocols." in *NDSS*, 2021.

[6] K. Shen, C. Wang, M. Guo, X. Zheng, C. Lu, B. Liu, Y. Zhao, S. Hao, H. Duan, Q. Pan *et al.*, "Weak links in authentication chains: A large-scale analysis of email sender spoofing attacks," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3201–3217.

[7] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," Tech. Rep., 2008.

[8] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," Tech. Rep., 2008.

[9] P. Gutmann, "Encrypt-then-mac for transport layer security (tls) and datagram transport layer security (dtls)," Tech. Rep., 2014.

[10] A. Langley, "A transport layer security (tls) clienthello padding extension," Tech. Rep., 2015.

[11] J. Mattsson and D. Migault, "Ecdhe_psk with aes-gcm and aes-ccm cipher suites for tls 1.2 and dtls 1.2," Tech. Rep., 2018.

[12] E. Rescorla, "The transport layer security (tls) protocol version 1.3," Tech. Rep., 2018.

[13] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, "Using frankencerts for automated adversarial testing of certificate validation in ssl/tls implementations," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 114–129.

[14] Y. Chen and Z. Su, "Guided differential testing of certificate validation in ssl/tls implementations," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 793–804.

[15] S. Y. Chau, O. Chowdhury, E. Hoque, H. Ge, A. Kate, C. Nita-Rotaru, and N. Li, "Symcerts: Practical symbolic execution for exposing noncompliance in x. 509 certificate validation implementations," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 503–520.