Advances and Applications in Mathematical Sciences
Volume 21, Issue 6, April 2022, Pages 3553-3562

A SECURITY MECHANISM TO DEFEND TLS CERTIFICATE AGAINST VULNERABILITIES

SURESH PRASAD KANNOJIA and JITENDRA KURMI

Department of Computer Science
University of Lucknow
Lucknow – 226007, India
E-mail: spkannojia@gmail.com
jitendrakurmi458@gmail.com

Abstract

In communication, Transport Layer Security certificates are mainly responsible for internet security. In the modern era, many web servers are vulnerable to man-in-the-middle (MITM) attacks by using forged but valid certificates that lead to privacy leakage and severe security risks. Forged certificates are those certificates which don't use proper validation standards. To solve this problem, we have proposed a Certificate Security Parameters Two-Phase Model (CSP-TPM), which is needed to deploy at the client-side to enforce stumble server to use CSP for formal validation and verification of TLS certificate for better security. We experimented two times, without the CSP-TPM model and with the proposed CSP-TPM model on an open-source dataset of Alexa top 1M domains. The experimental results and logical proof of correctness show that the servers with the proposed CSP-TPM model enhance the security to make communication robust from the forged but valid certificate.

1. Introduction

To identify inconsistencies in the Transport Layer Security (TLS) Libraries, automatic analysis of implementations is very prominent because of the completeness and safety-critical nature of the Secure Socket Layer (SSL)/TLS testing process for hostname verification [1]. Two main mechanisms for revoking certificates exist Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol. CRL provides downloadable lists of revoked certificates [2]. The Online Certificate Status Protocol (OCSP) allows clients to send short HTTP requests to their respective CA servers to check

2020 Mathematics Subject Classification: 68M12.
Keywords: X.509 Certificates, Certificate Authority, TLS, HTTPS, Certificate Transparency.
Received November 25, 2021; Accepted December 16, 2021

for revoked certificates [3]. Alternatively, OCSP stapling allows the servers to send revocation information within the initials handshake of TLS [4]. In literature, Standard PKI is used to implement X.509 that includes a certificate revocation list. This method only applies if customers have a modified copy of the CLR. Otherwise, web servers are vulnerable to a cyber-attack with revoked certificates [5].

But the major problem with CRL, OCSP, and Short-Lived certificates (SLC) is that they use web browsers to recognize and avoid fake certificates blocked by *C* As [6]. The evaluation and examination process of the TLS certificate error is done to support detailed analysis of TLS certificate verification and validation with certificate chain length [7, 8]. These methods were still subject to MITM attacks on the first domain connection. Cage uses the Top-Level Domain (TLD) analytics to limit the number of TLDs and is dependent on a CA to issue a TLS certificate [9]. Several guidelines for mitigating attacks on a domain certificate emitting approach and consumer reputation information were introduced [10, 11]. A formal security model is erupted to provide strong security guarantees against the keys reuse problem for multi-cipher suite key exchange protocols [12]. The servers must provide the SCT with the certificate during a TLS handshake if it supports the CT. Two additional components in CT mentioned are certificate monitoring and certified auditors. To check the accuracy of the public log auditor takes some data from the log and verifies with other information that the information is consistent [13].

During study, we found that the security of TLS certificates depends upon four parameters such as hostname verification, OCSP stapling, Certificate Chain length, and Certificate Transparency with signed certificate timestamps. But none of the single models is enough to defend the TLS certificates against the vulnerabilities with all above aspects (parameters). Hence it motivates us to design the CSP-TPM model to provide secure communication against the vulnerabilities by combining these parameters.

2. Methodology

Two phase methodology is used to implement the CSP-TPM model and analysed the Alexa 1 million domains. The first phase scans Alexa 1 million

domains for data collection using open-source SSLYZE, TLS client, and server responses were stored in JavaScript object naming (JSON) file for further analysis. The second phase analyse the server responses with proposed CSP-TPM and without CSPTPM and responses were observed based on above four parameters. We selected the Alexa domain that doesn't use four parameters to deliver SCT, known as non CSP domains. The proposed CSP-TPM model implemented using four algorithms such as HNV, OCSP stapling, CCL, and CT-SCT given below to enforce the CSP security to the client for the TLS certificate (See Figure.1).

Step 1. Start

Step 2. Select the non CSP domains from the first phase as input to the second phase.

Step 3. Fetch the server address and enforce the default CSP used by servers at stage 1. Examine the selected CSP having HNV or not. If the selected CSP does not have HNV then go to stage 2 and enforce the HNV only CSP to server otherwise terminate the process.

Step 4. Examine the selected CSP having HNV and OCSP or not. If the selected CSP does not have HNV and OCSP then go to stage 3 and enforce the HNV, OCSP only CSP to server otherwise terminate the process.

Step 5. Examine the selected CSP having HNV, OCSP and CCL or not. If the selected CSP does not have HNV, OCSP and CCL then go to stage 4 and enforce the HNV, OCSP, CCL only CSP to server otherwise terminate the process.

Step 6. Examine the selected CSP having HNV, OCSP, CCL and CT (SCT) or not. If the selected CSP does not have HNV, OCSP, CCL and CT (SCT) then go to stage 5 and enforce the HNV, OCSP, CCL, CT (SCT) only CSP to server otherwise terminate the process.

Step 7. If a certificate of domain contains all four CSP parameters then the system is highly secure.

Step 8. End

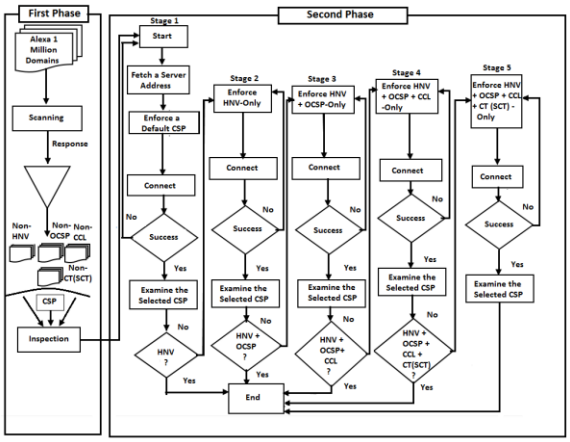

Figure 1. Proposed CSP-TPM Model.

Algo – 1. To verify and validate the Hostname Verification (HNV)

Step 1. Start

Step 2. Get the SSL/TLS certificate by cert = ssl_sock. get_peer_certificate()

Step 3. Decode the common name or Subject Name common_name = cert.get_subject().common Name. decode()

Step 4. Replacing the dot (.) and * from the common name to get the hostname regex = common_name.replace('.', r'\.'). replace ('*',r'.*') + '$'

Step 5. hostname validation process if re.matches(regex, host_name): #matches Output: Hostname matched else:#invalid Output: Hostname does not matched.

Step 6. Stop

Algo -2. To Verify and Validate OCSP stapling.

Step 1. start

Step 2. Load the request

Step 3. Rebuild an ocsp request into an object from encoded DER data ocsprequest = ocsp.loadder_ocsprequest(der_ocsprequest)

Step 4. Initializing the request certificate=

load.x509_certificate(pemcertificate) issuer = load.x509_certificate(pemissuer) buildocsp = ocsp.RequestBuilder() # SHA is in this example because RFC 5019 man- dates its use. buildocsp = addcertificate(certificate, issuer, SHA()) , req = buildocsp.build()

Step 5. Read the OCSP Responses from encoded DER data.

ocspresponse=ocsp.load.ocspresponse(ocspresponseunauth)

Step 6. Add the certificate status information into the OCSP request response.

certificate = load.x509_certificate(pemcertificate)

issuer = load.x509_certificate(pemissuer) responder certificate = load.x509_certificate(pemresponder certificate)

responderkey = serialization.load.private_key(pemresponderkey, None)

buildocsp = ocsp.ResponseBuilder()

SHA is in this example because RFC 5019 mandates its use.

buildocsp = addresponse(cert=cert, issuer=issuer, algorithm=hashes.SHA (), certificatestatus=ocsp.OCSPCerificatetStatus.GOOD, this_update=datetime.datetime.now(), next_update=datetime.datetime.now(), revocationtime=None, revocationreason=None, responder_id(

ocsp.OCSPResponderEncoding.HASH, re- sponder_cert response = buildocsp.sign(responderkey, hashes.SHA()) response.certificatestatus

Step 7. Stop

Algo – 3. To Verify and Validate Certificate Chain Length (CCL)

Step 1. Start

Step 2. Download and load the certificate from the url certificatestring = requests.get(certificateurl), certificate = crypto.loadcertificate(crypto. FILETYPE_PEM, str(certificatestring.text))

Step 3. Add your certificate to a trusted store by creating a certificate store.

store = crypto.X509Store()

Step 4. Let suppose certificates are stored in trusted-store in PEM format trusted_certs list for the certificate in trusted certificates:

store.add_certificate(certificate)

Step 5. Download and create a certificate context using the store

store_ctx = crypto.X509StoreContext(store, certificate)

Step 6. Check for certificate verification If (store_ctx.verify_certificate())

return True else return False

Step 7. Stop

Algo – 4. To verify and validate Certificate Transparency with Signed Timestamps (CST)

Step 1. Start

Step 2. Download the list of CT logs ctlogs = download_log_list()

Step 3. Finding the CT logs whose public key was used to sign the SCT

Step 4. Make a handshake to obtain the handshake response

handshake_res = do_handshake('google.com')

Step 5. Verify the CT logs with handshake response

verifications = verify_scts_by_tls(handshake_res, ctlogs)

Step 6. Using for loop to display the all verified handshake response

for ver in verifications: print('for{ver.verified}: {ver.log.description}')

True: Google 'Pilot' log True: Symantec log

Step 7. Stop

Proof of Correctness

For the mathematical proof of the proposed model, we formulated an equation (1) as per the combination of gates. We used logic gate and 16 possible functions of 4 input variables. In general, there is a 2^n function of n inputs. The AND Gate output goes HIGH to a logic level 1 only when all inputs are HIGH for all 4-inputs: HNV, OCSP stapling, CCL, and CT (SCT)

set to true that resulting in very secure. It means all CSP parameters are satisfied at a time. We use the summation notation to define the definite integral of a continuous function of one variable on a closed interval.

$$Svhs = \sum_{i=1}^{n} f(i) = fhnv(i) + focsps(i) + fccl(i) + fctsct(i) \quad (1)$$

Where $i = (1, 2, 3, 4)$, Svhs = Secure very high secure, and = Hostname verification, ocsps = online certificate status protocol stapling, ccl = certificate chain length, ctsct = certificate transparency signed certificate timestamp. If all values fhnv, focsps, fccl, and fctsct of equation (1), are set true, it means the system is highly secure.

3. Experimental Setup

The experiment has performed at the ICT Research Lab, University of Luck now, India, on a machine having Windows 10 operating system, core i3, 1.8 GHz processor, 8 GB of main memory equipped with 100 Mbps wired Ethernet cards with LAN connectivity. The dataset Open-Source Alexa 1 Million domains (dataset is available online [14]) is used. It performs first without the CSP- TPM model and second with the proposed CSP-TPM model. For this we wrote a python script based on an SSLYZE scanner to scan an SSL/TLS client to collect the data. A high-speed open-source TLS scanner can link simultaneously for performing TLS handshakes based on SSLYSE to the client [15].

4. Experimental Result and Analysis

Analysis of real-world certificate deployment on web servers based on number of certificates as per parameter out of 1 million Alexa sites. The analysis of common certificate error as per each parameter is, the hostname verification error without CSP-TPM is 9.10 %, and with proposed CSP-TPM, it reduces to 1.37 %. Similarly, the OCSP stapling error without CSP-TPM model is 7.90 %. But with proposed CSP-TPM model, it reduced up to 3.02 %. The certificate chain length error without CSP-TPM is 10.42%, which reduces to 3.14 % with proposed CSP-TPM model, and certificate transparency error is 32.58% without CSP-TPM model with SCT, which reduced to 14.83% with proposed CSP-TPM model. To calculate the percentage error of the same

certificate, we have uses the formula as given in equation (2) and results are tabulated in Table 1. The comparisons between without CSP-TPM and With Proposed CSP-TPM are visualised using bar chart visualisation technique shown in figure 2. Hence, we can enforce the new client side mechanism for TLS clients, which reduces the percentage of common certificate errors due to Hostname verification, OCSP stapling, Certificate chain length, and certificate transparency.

$$\text{Total Percentage} = \frac{\text{No. of observed certificate}}{\text{Total no. of certificate}} * 100\% \quad (2)$$

Table 1. Analysis of Certificate Parameters, without CSP-TPM, with proposed CSP-TPM and Common Certificate Error.

Certificate Parameters	No. of Certificate as per parameter out of (Alexa 1 million)	Without CSP-TPM		With Proposed CSP-TPM	
		No. of observed certificate	(Approx Error in %)	No. of observed certificate	(Approx Error in %)
HNV	69554	6329.41	9.10	952.88	1.37
OCSP stapling	60382	4770.17	7.90	1823.53	3.02
CCL	79644	8298.80	10.42	2500.79	3.14
CT(SCT)	256662	83620.47	32.58	38062.97	14.83

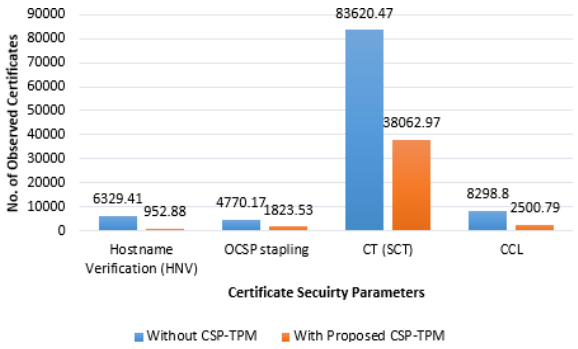

Figure 2. Comparison of number of Certificate Security Parameters without the CSP-TPM and with CSP-TPM model.

5. Conclusion and Future Scope

Certificates are the essential part of TLS to keep the communication secure and are used to identify valid certificates. The validity of the certificate depends on certificate authorities. The proposed CSP-TPM model encounters security flaws in forged but valid TLS certificates. The experimental result and proof of correctness show that the mechanism to enforce the stumble servers mandates, the use of CSP based parameters to reduce chance of significant security issues and privacy risks offered to clients. The result of reducing the certificate errors, overcome the vulnerabilities of Certificate System and advocate the trustworthiness of TLS certificates to their client. The CSP-TPM model performance and security will be enhanced by fine tuning in future.

References

[1] S. Sivakorn, G. Argyros, K. Pei, A. D. Keromytis and S. Jana, HVLearn: Automated black-box analysis of hostname verification in SSL/TLS implementations, In 2017 IEEE Symposium on Security and Privacy (SP) IEEE (2017), 521-538.

[2] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. T. Polk, Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280 (2008), 1-151.

[3] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, X. 509 Internet public key infrastructure online certificate status protocol-OCSP, (1999).

[4] Y. Pettersen, The transport layer security (TLS) multiple certificate status request extension, In RFC 6961 (2013, June).

[5] D. Basin, C. Cremers, T. H. J. Kim, A. Perrig, R. Sasse, and P. Szalachowski, Design, analysis, and implementation of ARPKI: an attack-resilient public-key infrastructure, IEEE Transactions on Dependable and Secure Computing 15(3) (2016), 393-408.

[6] T. H. J. Kim, L. S. Huang, A. Perrig, C. Jackson and V. Gligor, Accountable key infrastructure (AKI) a proposal for a public-key validation infrastructure, In Proceedings of the 22nd international conference on World Wide Web (2013), 679-690.

[7] 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), 114-129.

[8] 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), 793-804.

[9] J. Kasten, E. Wustrow and J. A. Halderman, Cage: Taming certificate authorities by inferring restricted scopes, In International Conference on Financial Cryptography and Data Security, Springer, Berlin, Heidelberg (2013), 329-337.

[10] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford and P. Mittal, Bamboozling certificate authorities with {BGP}, In 27th {USENIX} Security Symposium ({USENIX} Security 18) (2018), 833-849.

[11] M. Brandt, T. Dai, A. Klein, H. Shulman and M. Waidner, Domain validation++ for mitm-resilient PKI. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (2018, October), (pp. 2060-2076).

[12] X. Lan, J. Xu, Z. F. Zhang and W. T. Zhu, Investigating the multi-ciphersuite and backwards-compatibility security of the upcoming TLS 1.3, IEEE Transactions on Dependable and Secure Computing 16(2) (2017), 272-286.

[13] B. Laurie, A. Langley and E. Kasper, RFC6962: Certificate Transparency, Request for Comments IETF, (2013).

[14] Alexa Internet Inc 2021 Alexa Top-Sites, http://s3.amazonaws.com/alexa-static/top-1m.csv.zip, 2021. [Online; accessed 4-August-2021].

[15] Fast and powerful SSL/TLS scanning library, https://github.com/nabla-c0d3/sslyze, 2021, [Online; accessed 5-August- 2021].