United States Patent Application Publication	20250260583
Kind Code	A1
Publication Date	August 14, 2025
Inventor(s)	Zahradka; Trey J. et al.

AUTOMATIC DIGITAL CERTIFICATE REQUEST WITH SOFTWARE DEPLOYMENT

Abstract

A system includes a deployment server configured to receive an instruction to deploy software from an entity and in response to the instruction to deploy software, calling an application interface to request a new cryptographic certificate. An application server is configured to execute the application interface to receive the request for the new cryptographic certificate and in response to the request, obtain the new cryptographic certificate. A secure digital vault is configured to receive the new cryptographic certificate wherein the entity is not allowed to access the secure digital vault.

Inventors: **Zahradka; Trey J. (Blaine, MN), Colfer; Jeffrey M. (Raleigh, NC), Zirbes; Aaron J. (Minneapolis, MN)**

Applicant: **Target Brands, Inc.** (Minneapolis, MN)

Family ID: **1000007694383**

Appl. No.: **18/440352**

Filed: **February 13, 2024**

Publication Classification

Int. Cl.: **H04L9/32** (20060101); **G06F8/60** (20180101); **H04L9/30** (20060101)

U.S. Cl.:

CPC **H04L9/3263** (20130101); **G06F8/60** (20130101); **H04L9/30** (20130101);

Background/Summary

BACKGROUND

[0001] In asymmetric cryptography, an encryption algorithm applies data and an encryption key to a one-way function to produce encrypted data. A corresponding decryption algorithm applies the encrypted data and a decryption key to a second function to obtain the original data. The encryption key is made publicly available while the decryption key is kept secret. This allows anyone with the encryption key to encrypt data while only someone who has the secret decryption key can decrypt the data.

[0002] The encryption key is referred to as the public key and the decryption key is referred to as the private key. To use asymmetric cryptography, a first computer creates a public key/private key pair and then shares the public key while keeping the private key secret. If a second computer wants to send encrypted messages to the first computer, the second computer uses the public key to encrypt the message and then sends the encrypted message to the first computer. The first computer then uses the private key to decrypt the message.

[0003] One way to defeat asymmetric cryptography is to impersonate a server that a client wants to send encrypted information to. For example, a server can claim that it is a website www.spoof.net when it actually is not. The impersonating server then creates a public key/private key pair and sends the public key to unsuspecting client computers. The client computers use the public key to encrypt sensitive data and then send the encrypted data to the impersonating server. Since the impersonating server has the correct private key, the impersonating server is able to decrypt the sensitive data.

[0004] To prevent such spoofing, the Public Key Infrastructure (PKI) was created. Under this scheme, Certification Authorities certify that a public key is associated with a particular identity. To do this, Certificate Authorities issue a signed digital certificate, also known as a public key certificate or cryptographic certificate, that links a public key to an identity such as a hostname. When a client receives a signed digital certificate, the client verifies that the certificate was signed by a qualified certification authority and then assumes that the public key in the certificate is linked to the identity in the certificate.

[0005] The discussion above is merely provided for general background information and is not intended to be used as an aid in determining the scope of the claimed subject matter. The claimed subject matter is not limited to implementations that solve any or all disadvantages noted in the background.

SUMMARY

[0006] A method includes receiving an instruction to deploy software to a server and in response to the instruction, automatically requesting a new cryptographic certificate for the server.

[0007] In accordance with a further embodiment, a system includes a deployment server configured to receive an instruction to deploy software from an entity and in response to the instruction to deploy software, calling an application interface to request a new cryptographic certificate. An application server is configured to execute the application interface to receive the request for the new cryptographic certificate and in response to the request, obtain the new cryptographic certificate. A secure digital vault is configured to receive the new cryptographic certificate wherein the entity is not allowed to access the secure digital vault.

[0008] In a still further embodiment, a method includes receiving a request for a cryptographic certificate from a deployment server, wherein the deployment server is configured to deploy software to an application server in response to an instruction from an entity. The cryptographic certificate is generated and is stored in a secure digital vault that is accessible to the application server without exposing the cryptographic certificate to the entity.

[0009] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key

features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

Description

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. **1** is a block diagram of a system in accordance with one embodiment.

[0011] FIG. **2** is a flow diagram of a method in accordance with one embodiment.

[0012] FIG. **3** is an example of a Certificate Signing Request.

[0013] FIG. **4** is an example of a PEM certificate without a private key.

[0014] FIG. **5** is an example of a PEM certificate with a private key.

[0015] FIG. **6** is a block diagram of a computing device that implements the various embodiments.

DETAILED DESCRIPTION

[0016] Most cryptographic certificates, such as certificates under the X.509 standard, include an expiration date after which they are no longer valid. As a result, engineers responsible for maintaining a server must request a new certificate before the old certificate expires. Typically, this involves obtaining a public key/private key pair, using the public key to create a certificate signing request, sending the certificate signing request to a certificate authority to obtain a new signed certificate and passing the signed certificate and the private key to the server.

[0017] Such a system is less than ideal for a number of reasons. First, the system is not secure because the engineer is given access to the private key. This creates the possibility that the engineer will be able to decrypt messages that are sent to the server. Second, the system relies on the engineer remembering to request the certificate before the existing certificate expires. If the engineer forgets or is otherwise unable to request the new certificate, the server will stop functioning because cryptographic communication will no longer be allowed. Third, the system requires that an engineer perform a number of error-prone steps. In large scale environments where an engineer is responsible for a number of servers, this workload can be substantial.

[0018] In the embodiments provided below, a software deployment system is provided that automatically requests a new certificate for a server when software is deployed to the server. The system of the embodiments is more secure, more reliable and less labor intensive than prior art methods of keeping cryptographic certificates up to date.

[0019] FIG. **1** provides a block diagram of a system in accordance with one embodiment and FIG. **2** provides a method performed by the system of FIG. **1**.

[0020] In step **200** of FIG. **2**, a deployment instruction entity **100** issues a deployment instruction to a deployment server **102**. In accordance with one embodiment, the deployment instruction entity **100** is an engineer while in other embodiments, deployment instruction entity **100** is a software module.

[0021] The deployment instruction directs deployment server **102** to deploy an application or part of an application to an application server **104**. In accordance with one embodiment, the server is identified by a server cluster name and an environment (such as development or production) and the application is identified by an application identifier.

[0022] In step **202**, server preparation and software deployment modules **105** in deployment server **102** executes part of a pipeline associated with deploying code to the identified server. In particular, in step **202**, modules **105** call a certificate request evaluation module **106**, which determines whether automatic certificate requesting is enabled for the identified server. To make this determination, module **106** examines server cluster settings **108** to see if a New Certificate with Each Deployment setting **110** is set to True for the server cluster name and environment. If setting **110** is set to false, no new certificate is requested and the process of FIG. **2** ends at step **204**.

[0023] If setting **110** is set to true at step **202**, the last date and time **112** that a certificate was

requested for the server cluster and environment is retrieved from server cluster settings **108** at step **206**. The amount of time since the last certificate was requested is compared to a threshold time span at step **208**. If the amount of time is less than the threshold span, no new certificate is requested and the process of FIG. **2** ends at step **204**. The threshold span is selected as a balance between ensuring that the certificate will expire as late as possible while not wasting resources in obtaining a new certificate that will not extend the life span of the certificate significantly. In accordance with one embodiment, the time span is twenty-four hours.

[0024] After certificate request evaluation module **106** has determined that a new certificate should be requested at step **208**, server prep modules **105** call a certificate request module **114**, which calls a certificate API **116** executed on an application server **117** to request the new certificate.

[0025] In step **210**, certificate API **116** executes a key generator **118** to generate a private key/public key pair consisting of a private key **120** and a public key **122** that are used together during cryptographic operations. In particular, messages encrypted with the public key can be decrypted using the private key.

[0026] At step **212**, public key **122** and the name of the cluster server, the environment and the application identifier are used by a CSR generator **124** to generate a certificate signing request.

[0027] FIG. **3** provides an example of a decoded Certificate Signing Request **300** formed by CSR generator **124**. Certificate Signing Request **300** includes requestor attributes **302** and public key information **304**. Requestor attributes **302** include a country designation **306**, a state designation **308**, a locality designation **310**, an organization designation **312**, an organizational unit designation **314**, a common name designation **316** and an email designation **317**. Organization designation **312** provides the legal name of the organization requesting the Digital Certificate. Country designation **306**, state designation **308**, and locality designation **310** provide the country, state and city of the organization. Organizational unit designation **314** provides an internal name of a unit within the organization that is requesting the Digital Certificate. Common name designation **316** provides the address of the network server that the Digital Certificate is being issued for. Email designation **317** provides an email address to contact with questions about issuing the Digital Certificate.

[0028] Public key information **304** includes algorithm designation **318**, key length designation **320** and public key modulus and exponent designation **322** (also referred to as simply the public key designation). Algorithm designation **318** indicates the encryption algorithm that will be used during communications with the network server. Key length designation **320** indicates the length of the keys used in the certificate signing algorithm. Public key designation **322** provides the public key to be used during encrypted communications with the network server and consists of a modulus and an exponent.

[0029] In accordance with one embodiment, CSR generator **124** generates country designation **306**, state designation **308**, locality designation **310**, organization designation **312**, organizational unit designation **314**, and email designation **317** using preset values that are the same for each application server that deployment server **102** deploys code to. In other embodiments, one or more of these values may be provided in the call to certificate API server **116**. In accordance with one embodiment, common name designation **316** is constructed by CSR generator **124** using the cluster name and environment provided in the call to certificate API server **116** together with a preset domain. In FIG. **3**, the common name is set to <clustername>.<environment>.<PresetDomain>. For example, if the PresetDomain is “Target.com”, and the cluster name and environment that are passed in are TestCluster and Dev, respectively, the common name would become “TestCluster.Dev.Target.Com.”

[0030] CSR generator also uses preset values for the encryption algorithm and key length.

[0031] At step **214**, the generated certificate signing request **126** is used to request a signed certificate **128** from a certificate authority **130**. The signed certificate **128** provides assurances that public key **122** is associated with the entity responsible for application server **104**. Although certificate signing request **126** is shown being provided directly to certificate authority **130** in FIG.

1, in other embodiments, certificate signing request **126** is provided to an intermediary that selects a certificate authority. The selected certificate authority can be internal to an organization or external to the organization. One selection method is to select a preferred certificate authority set in a configuration file for one of the CSR attributes, such as the organizational unit or the common name, for example. A second selection method is to retrieve prices set in a configuration file for issuing Digital Certificates from the different certificate authorities in a selected collection of certificate authorities and then selecting the certificate authority with the lowest price. A third selection method is to use the limits on the number of Digital Certificates each certificate authority can issue to select the certificate authority. In accordance with one embodiment, the limits are used in conjunction with the number of Digital Certificates previously issued by each certificate authority to determine a number of certificates remaining for each certificate authority.

[0032] At step **216**, the signed certificate is received from certificate authority **130**. In one embodiment, the certificate is in the Privacy Enhanced Mail (PEM) format that includes the certificate but does not include the private key.

[0033] FIG. **4** provides an example of a signed PEM certificate **400**. Certificate **400** includes the signed certificate **402** in Base-64 encoding between a Begin Certificate header **404** and an End Certificate footer **406**. Signed certificate **402** includes the information provided in the Certificate Signing Request as well as a digital signature of the certificate authority. Because the private key is not provided to the certificate authority, PEM certificate **400** does not include the private key. The information in signed certificate **402** can be accessed by decoding signed certificate **402**.

[0034] At step **218**, private key **120** is added to the PEM certificate by a PEM modifier **132** to form a cryptographic file or modified PEM certificate **134**. FIG. **5** provides an example **500** of modified PEM certificate **134**. Example **500** includes signed certificate **402** between header **404** and footer **406**. In addition, example **500** includes Base-64 encoded private key **502** between Begin Private Key header **504** and End Private Key footer **506**. Encoded private key **502** includes information about the private key such as the modulus, the public exponent, the private exponent, and the two prime numbers used to form the modulus.

[0035] To provide for servers that use other certificate formats, a PFX and JKS producer **136** produces a PFX certificate **138** and a JKS certificate **140** at step **220**. PFX certificate **138** is in the Public-Key Cryptography Standards **12** format and includes the public-key certificate **128** and private key **120**. JKS certificate **140** is a Java KeyStore that includes public-key certificate **128** and private key **120**.

[0036] At step **222**, certificate API **116** returns modified PEM certificate **134**, PFX certificate **138** and JKS certificate **140** to deployment server **102**. Certificate storage module **142** in deployment server **102** then stores one or more of modified PEM certificate **134**, PFX certificate **138** and JKS certificate **140** in a secure storage vault **144**. Which of the returned files are stored and where they are stored in secure storage vault **144** are controlled by settings in server cluster settings **108**. After modified PEM certificate **134**, PFX certificate **138** and/or JKS certificate **140** are created, they are secured so that deployment instruction entity **100** is not exposed to the certificates or the private key within the certificates. Thus, deployment server **102** stores the one or more certificates in secure storage vault **144** while ensuring that deployment instruction entity **100** does not gain access to the certificates or private key **120**. For example, when the deployment instruction entity is an engineer, the engineer is prevented from seeing the certificate or private key.

[0037] At step **224**, the code that the deployment instruction entity wanted deployed to server **104** is sent to server **104** as deployed application **146**. In accordance with one embodiment, server **104** is provided with a path to the application to be deployed and then requests the file at that path location. Although step **224** is shown as occurring after the new digital certificate has been produced, in other embodiments, application **146** is deployed before the new certificate is requested. After application **146** has been deployed to server **104**, server **104** is restarted.

[0038] At step **226**, a startup routine **148** executed by application server **148** requests one of

certificates **134**, **138**, and **140** from secured storage vault **144**. Startup routine **148** parses the certificate retrieved from secure storage vault **144** to acquire private key **120** and certificate **128**. Application server **104** then provides certificate **128** to clients to allow those clients to communicate securely with application server **104**.

[0039] FIG. **6** provides an example of a computing device **10** that can be used as deployment server **102**, application server **104** or certificate API server **116**. Computing device **10** includes a processing unit **12**, a system memory **14** and a system bus **16** that couples the system memory **14** to the processing unit **12**. System memory **14** includes read only memory (ROM) **18** and random-access memory (RAM) **20**. A basic input/output system **22** (BIOS), containing the basic routines that help to transfer information between elements within the computing device **10**, is stored in ROM **18**. Computer-executable instructions that are to be executed by processing unit **12** may be stored in random-access memory **20** before being executed.

[0040] Embodiments of the present invention can be applied in the context of computer systems other than computing device **10**. Other appropriate computer systems include handheld devices, multi-processor systems, various consumer electronic devices, mainframe computers, and the like. Those skilled in the art will also appreciate that embodiments can also be applied within computer systems wherein tasks are performed by remote processing devices that are linked through a communications network (e.g., communication utilizing Internet or web-based software systems). For example, program modules may be located in either local or remote memory storage devices or simultaneously in both local and remote memory storage devices. Similarly, any storage of data associated with embodiments of the present invention may be accomplished utilizing either local or remote storage devices, or simultaneously utilizing both local and remote storage devices.

[0041] Computing device **10** further includes a solid state memory **25** and an optional hard disc drive **24**. Hard disc drive **24** is connected to the system bus **16** by a hard disc drive interface **32**. The drive and its associated computer-readable media provide nonvolatile storage media for the computing device **10** on which computer-executable instructions and computer-readable data structures may be stored. Other types of media that are readable by a computer may also be used in the exemplary operation environment as non-volatile memory such as solid-state memory.

[0042] A number of program modules may be stored in the drives and RAM **20**, including an operating system **38**, one or more application programs **40**, other program modules **42** and program data **44**. In particular, application programs **40** can include programs for implementing any one of modules discussed above. Program data **44** may include any data used by the systems and methods discussed above.

[0043] Processing unit **12**, also referred to as a processor, executes programs in system memory **14**, solid state memory **25** and disc drive **24** to perform the methods described above.

[0044] The computing device **10** may operate in a network environment utilizing connections to one or more remote computers, such as a remote computer **52**. The remote computer **52** may be a server, a router, a peer device, or other common network node. Remote computer **52** may include many or all of the features and elements described in relation to computing device **10**, although only a memory storage device **54** has been illustrated in FIG. **6**. The computing device **10** is connected to remote computer **52** through a network interface **60**.

[0045] In a networked environment, program modules depicted relative to the computing device **10**, or portions thereof, may be stored in the remote memory storage device **54**. For example, application programs may be stored utilizing memory storage device **54**. In addition, data associated with an application program may illustratively be stored within memory storage device **54**. It will be appreciated that the network connections shown in FIG. **6** are exemplary and other means for establishing a communications link between the computers, such as a wireless interface communications link, may be used.

[0046] Although elements have been shown or described as separate embodiments above, portions of each embodiment may be combined with all or part of other embodiments described above.

[0047] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms for implementing the claims.

Claims

1. A method comprising: receiving an instruction to deploy software to a server; in response to the instruction, automatically requesting a new cryptographic certificate for the server.
2. The method of claim 1 further comprising: in response to the instruction, generating a public key/private key pair and sending the public key when requesting the new cryptographic certificate.
3. The method of claim 2 wherein creating the public key/private key pair comprises preventing the private key from being exposed to any person.
4. The method of claim 3 further comprising: after receiving the new cryptographic certificate, storing the new cryptographic certificate in a secured location that the server can access but that no person can access.
5. The method of claim 4 further comprising: after storing the new cryptographic certificate, causing the server to access the secured location to retrieve the new cryptographic certificate.
6. The method of claim 5 wherein causing the server to access the secured location comprises causing the server to reboot such that the server requests the new cryptographic certificate during startup.
7. The method of claim 1 wherein automatically requesting the new cryptographic certificate comprises determining the last time a new cryptographic certificate was requested and automatically requesting the new cryptographic certificate when the last time is at least a threshold time span from the current time.
8. The method of claim 1 wherein automatically requesting a new cryptographic certificate for the server comprises determining that a setting for the server indicates that a new cryptographic certificate should be requested when software is deployed to the server.
9. A system comprising: a deployment server configured to: receive an instruction from an entity to deploy software; and in a response to the instruction to deploy software, calling an application interface to request a new cryptographic certificate; an application server configured to execute the application interface to: receive the request for the new cryptographic certificate; and in response to the request, obtain the new cryptographic certificate; and a secure digital vault, configured to receive the new cryptographic certificate, wherein the entity is not allowed to access the secure digital vault.
10. The system of claim 9 wherein the application server obtains the new cryptographic certificate by generating a public key/private key pair and submitting the public key to a certificate authority without exposing the private key to the entity.
11. The system of claim 10 wherein the application server obtains the new cryptographic certificate from the certificate authority and creates a certificate file by adding the private key to the new cryptographic certificate.
12. The system of claim 9 wherein before calling the application to request a new cryptographic certificate, determining that a time span since a current cryptographic certificate was created exceeds a threshold.
13. The system of claim 12 wherein before calling the application to request a new cryptographic certificate, determining that a setting for a second application server that is to use the new cryptographic certificate indicates that a new cryptographic certificate should be requested when software is deployed to the second application server.
14. The system of claim 13 wherein the second application server obtains the new cryptographic certificate and the private key from the secure digital vault.

15. A method comprising: receiving a request for a cryptographic certificate from a deployment server, wherein the deployment server is configured to deploy software to an application server in response to an instruction from an entity; obtaining the cryptographic certificate; and storing the cryptographic certificate in a secure digital vault that is accessible to the application server without exposing the cryptographic certificate to the entity.

16. The method of claim 15 wherein the request for the cryptographic certificate is generated by the deployment server in response to the instruction to deploy the software to the application server.

17. The method of claim 16 wherein the request for the cryptographic certificate is generated by the deployment server after determining that a time period since a last cryptographic certificate was requested exceeds a threshold.

18. The method of claim 17 wherein the request for the cryptographic certificate is generated by the deployment server after determining that a setting for the application server indicates that a new cryptographic certificate is to be requested when software is deployed to the application server.

19. The method of claim 15 wherein obtaining the cryptographic certificate comprises: generating a public key/private key pair without exposing the private key to the entity; and requesting a signed cryptographic certificate from a certificate authority.

20. The method of claim 19 further comprising adding the private key to the signed cryptographic certificate to form a cryptographic file in the secure digital vault.