BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

APPLICATION FOR MANAGING SSL CERTIFICATES
APLIKACE PRO SPRÁVU SSL CERTIFIKÁTŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ADAM JETMAR**
AUTOR PRÁCE

SUPERVISOR **Mgr. Ing. PAVEL OČENÁŠEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

156775

Institut:	Department of Information Systems (DIFS)
Student:	**Jetmar Adam**
Programme:	Information Technology
Title:	**Application for Managing SSL Certificates**
Category:	Networking
Academic year:	2023/24

Assignment:

1. Learn about SSL certificates, their structure and management. Familiarize yourself with existing certificate management applications.
2. Analyze requirements for a certificate management application. The application will provide basic operations with certificates, such as overview and management, filtering, expiration notifications, history change and management of certificate attributes.
3. Based on the results of the analysis and after consultation with the supervisor, design the application according to the specified requirements.
4. Implement the designed application.
5. Test the implemented application on real data.
6. Discuss the knowledge gained and suggest further extensions.

Literature:

- Kurose, J. F. Computer networking: A top-down approach. Pearson, Essex, 2017, ISBN 978-1-292-15359-9.
- Stallings, W. Network security essentials: Applications and standards. Hoboken, 2016, ISBN 978-0-13-452733-8.
- Bishop, M. Computer security: Art & Science. Addison-Wesley, Boston, 2003, ISBN 0-201-44099-7.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Očenášek Pavel, Mgr. Ing., Ph.D.**
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	30.10.2023

Abstract

This bachelor's thesis aims to develop a new application for managing SSL certificates within the organization of Red Hat. The primary goal of this development is to enhance the efficiency of SSL certificate management for Red Hat employees. The application enables centralized storage and processing of certificates, providing timely notifications for impending expirations. Additionally, it facilitates the creation of Jira tickets for necessary certificate renewals. Consequently, this initiative aims to mitigate application outages caused by expired certificates, providing employees with more effective solutions.

Abstrakt

Tato bakalářská práce si klade za cíl vytvořit novou aplikaci určenou pro správu SSL certifikátů v rámci organizace Red Hat. Hlavním cílem této vývojové iniciativy je optimalizovat efektivitu správy SSL certifikátů pro zaměstnance společnosti Red Hat. Aplikace umožňuje centralizované ukládání a práci s certifikáty a poskytuje včasné upozornění na blížící se expirace certifikátů. Kromě toho usnadňuje vytváření Jira ticketů pro nezbytné obnovení certifikátů. V důsledku toho má tato iniciativa za cíl zmírnit potíže spojené s výpadky aplikací způsobenými expirovanými certifikáty, poskytujíc zaměstnancům účinnější řešení.

Keywords

JavaScript, React, Node.js, Wasp, Linux, OpenShift, SSL, Jira, Red Hat

Klíčová slova

JavaScript, React, Node.js, Wasp, Linux, OpenShift, SSL, Jira, Red Hat

Reference

JETMAR, Adam. *Application for Managing SSL Certificates.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Mgr. Ing. Pavel Očenášek, Ph.D.

Application for Managing SSL Certificates

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Mgr. Ing. Pavel Očenášek, Ph.D. The supplementary information was provided by Mr. Bc. Nicholas Forrer. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Adam Jetmar
January 29, 2024

Acknowledgements

I express my sincere appreciation to Mr. Mgr. Ing. Pavel Očenášek, Ph.D., for his guidance and support during this thesis. I also acknowledge Mr. Bc. Nicholas Forrer for his technical insights, contributing to the development process. Their invaluable contributions were crucial to the successful completion of this work.

Contents

List of Figures

Chapter 1

Introduction

In our rapidly evolving digital era, the safeguarding of online communications has become increasingly crucial, with Secure Sockets Layer (SSL) certificates playing a pivotal role. This Bachelor's thesis, entitled „Application for Managing SSL Certificates," targets this essential aspect, with a specific lens on Red Hat's unique environment, a frontrunner in the realm of open-source technology. The impetus for this study is the growing prominence of SSL certificates in ensuring secure internet interactions. As our digital world continues to expand and transform, there is a pressing demand for more advanced, efficient, and user-friendly systems to oversee these certificates. This thesis aims to create an application focused on the effective management of SSL certificates, specifically crafted to align with Red Hat's operational needs and context. The goal is not only to devise a practical tool but also to contribute meaningfully to the broader conversation on digital security and certificate management.

The thesis is structured to offer an in-depth understanding of the subject matter. The second chapter establishes the theoretical groundwork, delving into the essentials of SSL certificates. This includes examining the historical relevance and significance of SSL certificates, their technical aspects and security norms, and a survey of existing solutions in SSL certificate management. The third chapter undertakes an analysis of Red Hat's particular requirements, scrutinizing both the functional and security necessities, the adaptation of the application to align with company processes, and its interaction with existing systems and infrastructure.

The fourth chapter transitions to the design of the application, elaborating on its architectural framework, the selection process for technologies and tools, and the overall plan for implementation. The fifth chapter covers the actual implementation, discussing the various stages of development, the methodological approach, challenges and resolutions encountered, and the integration of the application within Red Hat's infrastructure.

The sixth chapter is centered around testing and validation, detailing the strategy and blueprint for testing, examining the features and security facets, and evaluating the outcomes and necessary refinements. The seventh chapter contemplates the application's contribution to Red Hat, exploring potential avenues for expansion and further development, and offering recommendations for its effective utilization.

The thesis culminates with the eighth chapter, summarizing the principal accomplishments, contemplating future research prospects, and presenting concluding remarks. This structured approach ensures that the thesis not only tackles a significant challenge in digital security but also presents a tailored solution for a leading technology corporation.

Chapter 2

SSL certificates

2.1 SSL/TLS certificates

SSL/TLS certificates, also known as x.509 certificates, are digital documents signed by Certificate Authorities (CAs), serving as validation for the ownership of a public key. These certificates confirm the legitimacy of servers or clients and are integral to public key cryptography. Major operating systems and browsers contain built-in trust stores of these CAs, facilitating automatic trust in certificates issued by them. The presence of a valid SSL/TLS certificate on a webpage is indicated by 'https' in the website's URL, and in some cases, an Extended Validation certificate can turn the HTTPS padlock green, offering additional assurance of the website's authenticity.

2.2 History and significance of SSL certificates

The historical trajectory of Secure Sockets Layer (SSL) certificates epitomizes a significant evolution in the field of internet security, mirroring the rapid changes and demands of the digital world. This section aims to explore the genesis and development of SSL certificates, highlighting their indispensable role in contemporary online security and communication.

Originating in the mid-1990s, SSL certificates were a groundbreaking innovation for transmitting information securely over the internet. Initially conceptualized to protect sensitive data like credit card information and passwords, these certificates swiftly became fundamental in establishing online trust, confidentiality, and encryption. The genesis of SSL is attributed to the pioneering work at Netscape, specifically by Taher Elgamal, an Egyptian cryptographer, who played a key role in developing the first SSL internet protocol.

The late 1990s were a formative period for SSL. The first version, SSL 1.0, was beset with cryptographic weaknesses and was not released. Its successor, SSL 2.0, released in 1995 with Netscape Navigator, was quickly superseded due to security concerns. Concurrently, Microsoft's development of the PCT (Private Communication Technology) protocol did not achieve widespread acceptance.

The advent of SSL 3.0 marked a significant improvement in stability and security, paving the way for the emergence of TLS (Transport Layer Security). In 1999, TLS 1.0 was introduced, predominantly as a refined version of SSL 3.0, under the guidance of the IETF (Internet Engineering Task Force). This was a critical juncture, reflecting a collective effort towards standardizing internet security protocols.

As the 21st century commenced, SSL certificates were relatively rare and expensive, predominantly offered as Business Validation certificates by Certificate Authorities. This scenario began to shift in 2002 with GeoTrust's introduction of Domain Validation certificates, democratizing access to encryption. The 2007 introduction of Extended Validation certificates further strengthened online transaction security and effectively reduced phishing threats.

The most recent decade in SSL certificate history has been significantly influenced by tech giants and emerging industry standards. In 2014, Google's decision to favor secure websites in search engine rankings dramatically increased HTTPS adoption. This trend was further bolstered by Cloudflare's distribution of free certificates and the advent of the Let's Encrypt authority, broadening the accessibility of SSL/TLS.

Throughout these phases, SSL protocols have been refined to bolster security measures and address emerging vulnerabilities. The 2018 release of TLS 1.3, for instance, marked an advancement in encryption protocols and efficiency. These continuous enhancements underscore the commitment to enhancing internet security.

The journey of SSL/TLS protocols, from their inception to becoming a cornerstone of web security, exemplifies the dynamic nature of cybersecurity and the industry's adaptability to evolving threats. As the digital environment continues to evolve, the significance of SSL certificates in maintaining data integrity and establishing digital trust remains paramount.

The information presented is derived from data obtained from the website [2].

2.3 Technical aspects and security standards

SSL, the precursor to TLS, and TLS itself are cryptographic protocols that endow web communications with essential security features, ensuring data integrity and resilience against unauthorized access. Introduced in 1999, TLS supersedes SSL due to its enhanced security measures, despite SSL's nomenclature often being used generically to describe both. Modern systems exclusively support TLS, with SSL being fully deprecated. Connections secured via TLS are denoted by HTTPS (Hypertext Transfer Protocol Secure) in web browser address bars, contrasting the unsecured HTTP.

The necessity of TLS arises from the inherent vulnerability of web communications without such protection. Unsecured connections, such as a user logging into an online store, are susceptible to eavesdropping, where sensitive information can be intercepted. TLS serves as a fundamental tool for end-to-end encryption, ensuring data privacy and verifying the authenticity of the communicating parties. Its deployment has become a global standard, with leading web browsers mandating secure TLS connections for website access.

The information presented is derived from data obtained from the website [1].

2.3.1 The SSL/TLS handshake

The TLS handshake is a critical process where two systems verify each other's compatibility with TLS, agreeing on encryption algorithms[1] and cipher suites[2]. This handshake facilitates the establishment of a secure communication line. The steps involve:

[1]https://www.appviewx.com/education-center/what-is-sha-and-what-are-sha-1-and-sha-2/
[2]https://www.appviewx.com/education-center/cipher-suites/

1. The user begins the handshake by sending a “Hello” message, which includes the supported TLS protocol, preferred cipher suites, and a unique “client random” byte string.
2. The server replies with its SSL certificate, the chosen cipher suite for the session, and its own “server random” byte string.
3. The user’s browser verifies the server’s SSL certificate and its issuing authority, confirming the server’s authenticity.
4. The client sends another message (the premaster secret) which is encoded with the SSL certificate’s public key.
5. The server decrypts this message with its private key.
6. Both parties use the “client random,” “server random,” and the premaster secret to generate a session key for encrypted communication.
7. Both server and client exchange “finished” messages encrypted with the session key, signaling the completion of the handshake.
8. The handshake concludes, enabling both parties to communicate securely using the established session keys.

The SSL/TLS handshake process information was sourced from SSL2Buy’s website [4].

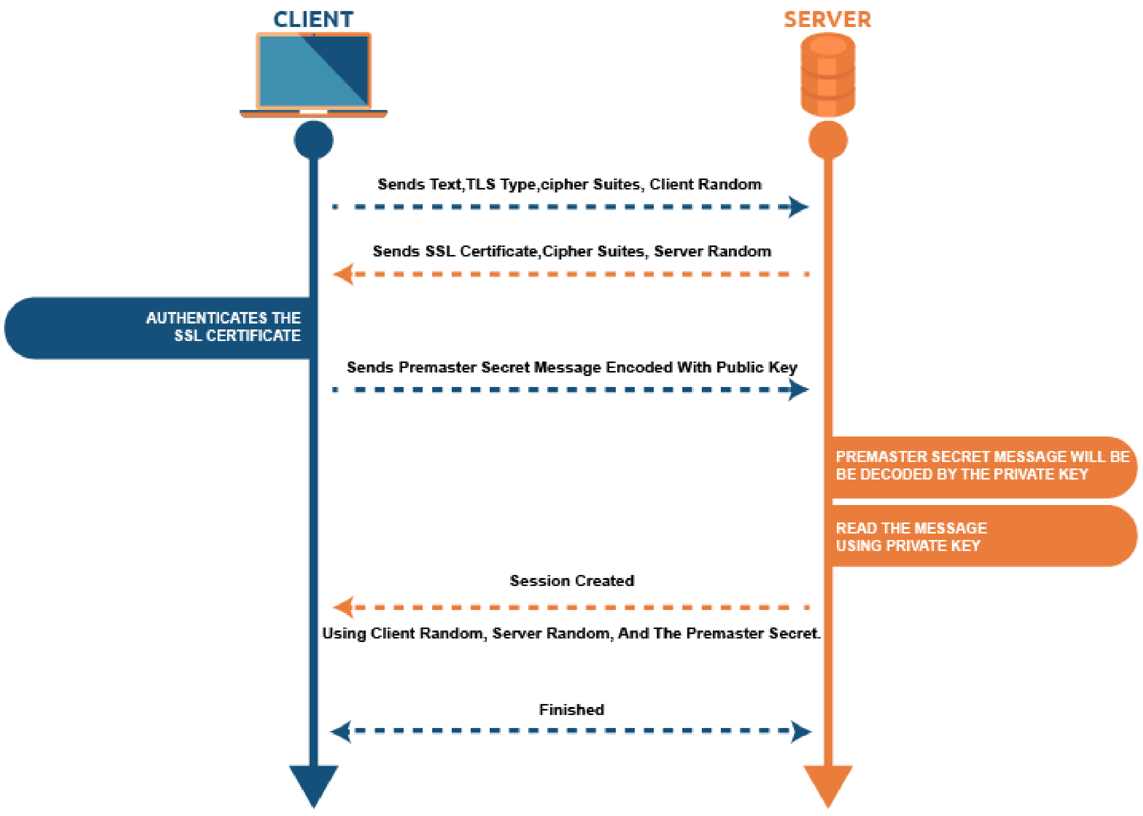
CLIENT
SERVER
Sends Text,TLS Type,cipher Suites, Client Random
Sends SSL Certificate,Cipher Suites, Server Random
AUTHENTICATES THE SSL CERTIFICATE
Sends Premaster Secret Message Encoded With Public Key
PREMASTER SECRET MESSAGE WILL BE BE DECODED BY THE PRIVATE KEY
READ THE MESSAGE USING PRIVATE KEY
Session Created
Using Client Random, Server Random, And The Premaster Secret.
Finished

Figure 2.1: SSL/TLS handshake process

The image was sourced from SSL2Buy’s website [4].

2.3.2 Assymetric vs. symmetric encryption

TLS employs both asymmetric and symmetric encryption. Asymmetric encryption, involving a public-private key pair, is used initially to establish the secure connection and generate a session key. However, due to its resource-intensive nature, symmetric encryption, which uses a shared key for both encryption and decryption, is employed for ongoing communication. This combination ensures a balance of security and efficiency in data transmission.

The information presented is derived from data obtained from the website [5].

2.4 Overview of current solutions for SSL certificate management

The 2021 State of Machine Identity Management Report sheds light on a critical aspect of digital security management, revealing that around 40% of organizations continue to manage their SSL/TLS certificates manually, often using basic tools like spreadsheets. This traditional approach has been linked to an average of three unforeseen disruptions over a period of two years, primarily due to certificate expiration or configuration errors.

Despite advancements in digital infrastructure and the adoption of agile computing systems, a considerable segment of the corporate world still adheres to these outdated certificate management practices. Such manual methods are increasingly inadequate for the dynamic and rapid demands of current digital ecosystems. The primary concern with these manual techniques is that they introduce a significant risk of operational interruptions and compliance violations, largely attributed to human inaccuracies.

2.4.1 Automated certificate management environments

The Automated Certificate Management Environment (ACME) protocol is a critical development in the realm of digital certificate management, particularly within the framework of Public Key Infrastructure (PKI).

Origins and development of ACME

Initially developed by the Internet Security Research Group (ISRG) for its Let's Encrypt initiative, ACME has emerged as a revolutionary protocol designed for the automated issuance and renewal of digital certificates, significantly reducing the need for manual intervention. Let's Encrypt, notable for its provision of complimentary domain validated (DV) certificates, has propelled ACME to widespread recognition and adoption across various Certificate Authorities (CAs) and PKI vendors.

ACME experienced a notable evolution with its first version (ACME v1) in 2016, primarily facilitating single-domain certificate issuance. The subsequent iteration, ACME v2, introduced in 2018, expanded its capabilities to include wildcard certificates and augmented domain ownership verification methodologies. The International Engineering Task Force (IETF) standardized ACME with RFC8555 in 2019, marking a pivotal moment in the protocol's evolution and adoption.

Mechanics of ACME operations

ACME's operational framework is centered on automating complex and time-intensive tasks like CSR generation, domain ownership verification, certificate issuance, and installation.

Primarily used for obtaining DV certificates, ACME can also be employed for more advanced certificates such as OV and EV, given appropriate additional mechanisms are in place.

The protocol operates via a client-server model, with the ACME client located on the server or device requiring a trusted SSL/TLS certificate, communicating with an ACME server maintained by a CA. This communication, structured around JSON messages over HTTPS, encompasses various certificate management actions, including issuance and revocation.

ACME implementation

The deployment of an ACME client involves selecting an appropriate client, installing it on the targeted domain or server, followed by a sequence of configuration steps. These include the selection of a CA supporting ACME, domain specification, authorization key pair generation, and completion of domain control challenges. The CA then issues a nonce, which the client signs with its private key to demonstrate domain control.

In practice, ACME significantly streamlines the issuance and renewal of certificates. An ACME agent on the web server requests a certificate via CSR to the CA, which, upon validating the CSR and the domain's authorized key, issues the certificate. Revocation of certificates follows a similar automated pathway, enhancing the efficiency and security of the certificate management process.

ACME's impact on automation

Despite the advent of agile computing, manual certificate management remains prevalent, often leading to inefficiencies and security risks. ACME addresses these issues by automating domain control verification and certificate lifecycle management, thereby mitigating the risks associated with manual errors and certificate misconfigurations.

CA agility facilitated by ACME

The concept of CA agility, referring to the ease of switching between CAs, is an industry best practice. ACME contributes to this agility by simplifying the process of choosing alternative CAs, thus providing a safeguard against potential compromises or outages associated with dependency on a single CA.

Integration of ACME in Keyfactor

Platforms like Keyfactor[3] have integrated ACME to provide comprehensive automation and self-service capabilities in certificate management. By functioning as an ACME server, such platforms ensure efficient, error-minimized management across the entire certificate lifecycle.

The information presented is derived from data obtained from the website [6].

2.5 Certificate lifecycle management tools

Certificate Lifecycle Management (CLM) and PKI software are essential in the cryptography framework that underpins the security of digital communication. This framework plays a crucial role in safeguarding data, devices, and identities against a spectrum of cyber threats,

[3]https://www.keyfactor.com/

including impersonation, unauthorized interception, and data tampering. The primary advantage of these software solutions lies in their ability to provide comprehensive visibility and automation across the entire certificate lifecycle. This lifecycle encompasses a series of critical stages, including certificate issuance, discovery, inventory, provisioning, deployment, security enhancement, monitoring, renewal, and, ultimately, revocation.

Traditionally, the management of digital certificates has been conducted through manual methods, such as spreadsheet tracking. However, CLM and PKI software offer a significant advancement by automating these processes, thereby reducing the risks of unplanned system downtime and vulnerabilities that may arise due to human errors or overlooked certificate expirations.

Capabilities and Use Cases of CLM and PKI Software

These software solutions are designed to handle various types of digital certificates, extending beyond SSL and TLS certificates to include client authentication, digital signature, and SSH certificates, among others. The broad application of PKI and CLM software covers diverse use cases. These range from user and machine-to-machine authentication (critical for servers and containers) to digital code and document signing, as well as ensuring the encryption and integrity of IoT devices.

To be considered a viable solution in the Certificate Lifecycle Management and PKI category, a product must exhibit specific characteristics and capabilities:

1. Automation of Certificate Lifecycle Management: This includes the automation of processes such as discovery, inventory, provisioning, deployment, securing, monitoring, renewal, and revocation. Some solutions may also offer certificate issuance capabilities through public certificate authorities or private PKI functionalities.

2. Centralized Control and Visibility: The software should provide a centralized platform for visibility and control, allowing comprehensive management and reporting on certificates, keys, and ciphers.

3. Monitoring and Notification Features: It is imperative for these tools to monitor and alert administrators about impending certificate expirations. Additionally, they should possess workflows to automatically execute specified actions like certificate renewal or revocation.

4. Multi-CA Support: The ability to support certificates from multiple Certificate Authorities (CAs) is a crucial criterion, ensuring versatility and broad applicability.

The information presented is derived from data obtained from g2's website [8].

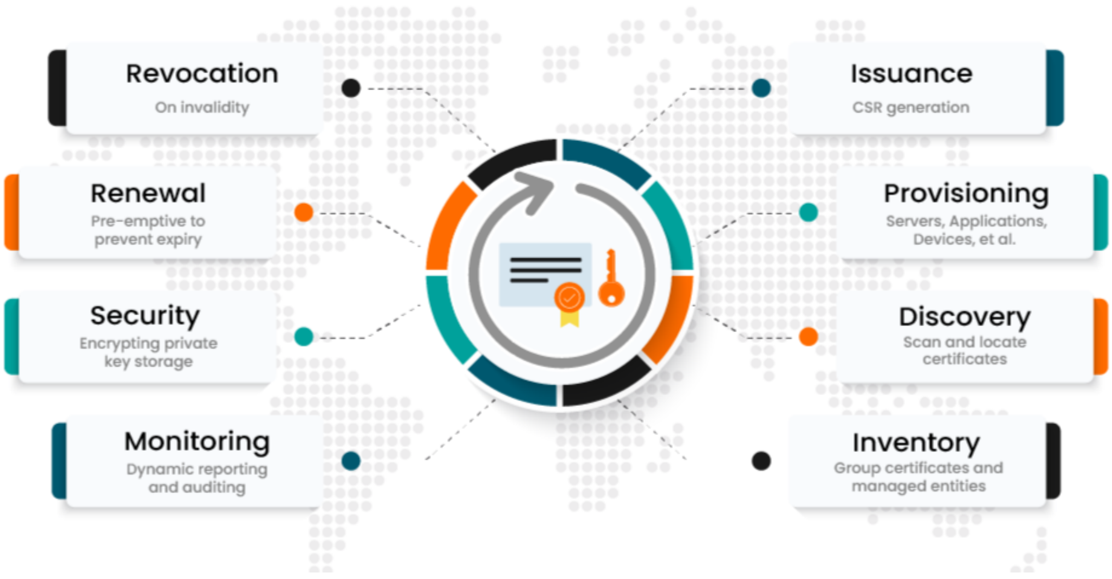
Revocation
On invalidity
Issuance
CSR generation
Renewal
Pre-emptive to prevent expiry
Provisioning
Servers, Applications, Devices, et al.
Security
Encrypting private key storage
Discovery
Scan and locate certificates
Monitoring
Dynamic reporting and auditing
Inventory
Group certificates and managed entities

Figure 2.2: Stages of certificate lifecycle management

The image was sourced from appviewx's website [7].

2.6 Challenges in SSL Certificate Management

In the exploration of SSL certificate management, several challenges emerge, particularly in the context of evolving digital infrastructures and the increasing reliance on encryption.

The shift from traditional data centers to a cloud-centric approach and the sharp adoption of digital devices have significantly impacted the domain of cryptography, upon which PKI is dependent. A notable industry survey highlighted that over 80% of an organization's traffic is encrypted, a trend rising against the backdrop of cloud adoption and remote work policies. This transformation necessitates a robust and adaptable approach to certificate management.

2.6.1 Threats Associated with Certificate Management

Two primary threats are inherently connected to certificates if not managed adeptly: system outages and security breaches. These challenges intensify as the number of connected devices and users within an organization grows. The complexities involved in issuing, deploying, and revoking certificates for each device and application are compounded by the need to prevent unauthorized certificate requests, manage the certificates' lifecycle, and protect against potential abuses in cyber-attacks.

2.6.2 Inefficiencies in Traditional Management Practices

Despite the critical nature of certificate management, many organizations still resort to reactive and rudimentary tools, such as PowerShell scripts or Excel spreadsheets, for tracking and managing certificates. These methods, while familiar, are prone to human error and inefficiency. An apt analogy is the potential network-wide impact due to the failure to renew or update a certificate on a single server or application. A minor oversight could escalate into a significant operational disruption. An illustrative example of this was the outage experienced by Microsoft's collaboration platform, Teams, where an overlooked certificate renewal on one server led to the platform's temporary shutdown.

The information presented is derived from data obtained from zensar's website [3].

Chapter 3

Analysis of Red Hat's Requirements

In addressing the specific needs of Red Hat[1] for SSL certificate management, a comprehensive analysis reveals the necessity for an advanced system that combines both functional efficiency and robust security measures. This system is required to adeptly navigate the complexities of certificate management in a rapidly evolving digital environment.

3.1 Functional and Security Requirements

At the forefront of the system's capabilities is its proficiency in monitoring the expiration of various types of certificates, including those issued by Digicert, self-signed, and AuthZ. This tracking feature is integral for maintaining an up-to-date certificate inventory. Equally important is the system's ability to present a holistic view of all tracked certificates, encompassing essential details such as the Certificate's Common Name (CN), associated contact information, expiration date, type, and the issuing Certificate Authority (CA).

The deployment of this system necessitates a streamlined and effective pipeline, ensuring seamless integration and deployment within Red Hat's existing infrastructure.

An additional feature of paramount importance is the ability to efficiently remove or discontinue tracking of certificates that are no longer required, thereby maintaining an optimized and relevant certificate inventory.

A critical functionality of the proposed system is its sorting mechanism, specifically the ability to organize certificates according to their expiration dates. This feature aids significantly in prioritizing certificate renewal and management activities. Complementing this is the need for a mechanism to label certificates as outdated, effectively removing them from active visibility in the user interface, a crucial step for maintaining governance and control over the certificate environment.

The renewal and updating of certificate data also form a core component of the system's requirements. The system should facilitate the uploading of updated certificates or the marking of a certificate as renewed to prevent misidentification as expired, particularly for manually uploaded certificates. This capability works in tandem with an automated tracking feature that manages the lifecycle of certificates.

Tracking certificates via URLs represents an innovative approach, allowing the system to monitor the primary certificate of a given website. In addition, the system should

[1] https://www.redhat.com/en

include a tagging feature for associating certificates with specific applications, enhancing the organizational structure and visibility within the user interface.

A robust notification system is essential, particularly for certificates nearing expiration. The system should be configured to send timely alerts as certificates approach their expiration dates, following a structured timeline that includes intervals of 30 days, 15 days, 5 days, and post-expiration. The user interface should support this through a color-coded system, employing distinct colors to indicate the urgency of certificate renewal.

Given the operational context of Red Hat, the system must function within the confines of Red Hat's VPN, ensuring secure and protected access. Accessibility to essential documentation for certificate renewal and CA information directly through the system is also a requisite, providing users with immediate access to necessary resources.

The tracking of historical changes made to certificates is another key requirement, offering transparency and traceability. Integration with project management tools such as Jira[2] for automated creation of renewal tickets further augments the system's capabilities. Additionally, the system should support the automatic generation of Certificate Signing Requests (CSRs) and Keys, simplifying the process of certificate issuance.

The implementation of Single Sign-On[3] (SSO) is crucial for efficient user authentication. Moreover, the system must allow for the creation of team-specific access to certificates based on Rover Groups, enabling users to access only those certificates pertinent to their team, while also incorporating an administrative role to manage access across different teams.

In conclusion, the implementation of a certificate management system for Red Hat calls for a multifaceted approach that addresses both functional and security aspects. This system should not only efficiently manage the lifecycle of certificates but also ensure adherence to Red Hat's specific security protocols and operational practices.

3.2 Interaction with Existing Systems and Infrastructure

The seamless integration of this application with the current infrastructure is essential for ensuring its effectiveness and alignment with Red Hat's operational strategies.

3.2.1 Deployment on OpenShift Platform

The application is tailored to be deployed on Red Hat's OpenShift[4], a sophisticated containerization platform. This choice of infrastructure is strategic, leveraging OpenShift's strengths in scalability, security, and efficient application management. The integration with OpenShift is instrumental in harnessing these benefits, thereby enhancing the application's performance and reliability.

3.2.2 Utilization of GitLab CI/CD for Automation

A key feature of the application is its incorporation of GitLab's CI/CD pipelines, aiming to automate the certificate renewal process. By integrating with GitLab's pipelines, the application streamlines the renewal and management of SSL certificates, reducing manual processes and improving operational efficiency. This automation is crucial in maintaining the integrity and timeliness of certificate management.

[2]https://www.atlassian.com/software/jira

[3]https://www.techtarget.com/searchsecurity/definition/single-sign-on

[4]https://www.redhat.com/en/technologies/cloud-computing/openshift

3.2.3 Incorporation of Red Hat SSO for Authentication

The application employs Red Hat Single Sign-On (SSO), which uses Keycloak[5] for identity and access management, facilitating secure and efficient user authentication. The integration process involves linking the application with Red Hat SSO via a provided URL and public certificate, ensuring secure connectivity. Post-integration, user authentication is managed through Red Hat SSO, with a seamless user redirection mechanism between the SSO and the certificate management application.

3.2.4 Integration with Rover Groups for Team Management

Adopting a team-centric approach, the application aligns with Rover Groups, an internal team management system. This integration allows for precise and organized allocation of certificates to relevant teams, ensuring that certificate access and management are tailored to specific team requirements. This feature plays a vital role in maintaining a structured and efficient certificate management environment within the organization.

3.2.5 Jira API Connectivity for Automated Ticketing

The application is configured to interface with Jira's API, enabling automatic creation of tickets for certificate renewal tasks. This integration ensures that renewals are systematically tracked and managed within Red Hat's existing project management tools. Automating the ticketing process through Jira API integration is a significant step towards enhancing the responsiveness and accuracy of the certificate renewal process.

[5] https://www.keycloak.org/

Bibliography

[1] appviewx. *What is TLS/SSL Protocol?* [online]. 30. december 2023 [cit. 30.12.2023]. Available at: https://www.appviewx.com/education-center/what-is-tls-ssl-protocol/.

[2] Gitlan, D. *SSL History: From Inception to Evolutionary Triumph* [online]. 28. december 2023 [cit. 28.12.2023]. Available at: https://www.ssldragon.com/blog/history-of-ssl/.

[3] Kamboj, P. *Certificate Management – Challenges and How to Manage Effectively* [online]. 17. january 2024 [cit. 17.1.2024]. Available at: https://www.zensar.com/insights/blogs/certificate-management-challenges-and-how-manage-effectively/.

[4] ssl2buy.com. *SSL/TLS Handshake: Detailed Process and How does it Work* 1. . [cit. 13.1.2024]. Available at: https://www.ssl2buy.com/wiki/ssl-tls-handshake-how-does-it-work.

[5] Team, S. S. *What is SSL/TLS: An In-Depth Guide* [online]. 30. december 2023 [cit. 30.12.2023]. Available at: https://www.ssl.com/article/what-is-ssl-tls-an-in-depth-guide/.

[6] Thompson, M. *What is ACME protocol and how does it work?* [online]. 6. january 2024 [cit. 6.1.2024]. Available at: https://www.keyfactor.com/blog/what-is-acme-protocol-and-how-does-it-work/.

[7] www.appviewx.com. *A comprehensive how-to guide for certificate lifecycle management (CLM)* [online]. 14. january 2024 [cit. 14.1.2024]. Available at: https://pages.appviewx.com/rs/249-TWN-899/images/A%20Comprehensive%20How-to%20Guide%20to%20CLM.pdf.

[8] www.g2.com. *Best Certificate Lifecycle Management (CLM) Software* [online]. 14. january 2024 [cit. 14.1.2024]. Available at: https://www.g2.com/categories/certificate-lifecycle-management-clm.