International Journal for Modern Trends in Science and Technology, 6(12): 248-252, 2020
Copyright © 2020 International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST061245
Available online at: http://www.ijmtst.com/vol6issue12.html

Automated SSH Key Management Using Custom Certificate Authority and Temporary Keys

Kartik Behl[1] | Amita Goel[1] | Vasudha Bahl[1] | Nidhi Sengar[1]

[1]Information Technology, Maharaja Agrasen Institute of Technology

To Cite this Article
Kartik Behl, Amita Goel, Vasudha Bahl and Nidhi Sengar, "Automated SSH Key Management Using Custom Certificate Authority and Temporary Keys", *International Journal for Modern Trends in Science and Technology*, 6(12): 248-252, 2020.

Article Info
Received on 10-November-2020, Revised on 02-December-2020, Accepted on 06-December-2020, Published on 10-December-2020.

ABSTRACT

In this era of massive knowledge, cloud computing has emerged as one of the most on-demand accessibility of computing system resources, a plan that modified the ways of computing. However, data security remains a tangle in creating cloud computing vulnerable. Therefore, SSH users and businesses victimization SSH coding should do everything necessary to shield the safety of their SSH encryption keys and different components so as to uphold the trust placed within the system. SSH keys automate the manner of stable get admission to servers, bypassing the want to manually input log-in credentials. SSH is likewise immune to brute pressure assaults and protects against sure assault vectors getting used to advantage get admission to far off machines. The current research work carries out a detailed functioning of a publicly available, fully managed and automated solution for secure management and distribution of SSH keys with a granular level access control which will prevent any potential leak of a private key.

KEYWORDS: *Cloud Computing, SSH, Private Key*

I. Introduction

Servers largely contain proprietary applications and sensitive data which can't be risked obtaining leaked because it will cause an incredible loss of revenue, an enormous quantity of overspending and might even result in a suit. Hence, making security a superlative priority and SSH could be a total answer to permit trusty, encrypted connections to different systems, networks, and platforms, which might be remote, within the information cloud, or distributed across several locations. It replaces separate security measures that antecedent were accustomed cypher information transfers between computers.

In particular, Secure Shell (SSH) is a cryptographic system protocol utilized by many entities, including businesses and organizations, to offer stable records communication, faraway get entry to and control, and different stable community offerings to people related to the entity however bodily far away from the networked devices. In general, SSH structures rent public keys saved at the SSH server that can be used to authenticate the data that could handiest be supplied via way of means of a holder of the non-public key similar to the general public key.

Now and again, in light of the fact that the public keys are enrolled with the SSH gadgets, the authorized key documents become gigantic and asset concentrated. To boot, as people leave or break up with the entity the general public keys generally stay at intervals the licensed key file because the public keys aren't mapped to individual users. Now and again, this outcomes in unclaimed public keys dwelling in the approved key record or in people holding unapproved admittance to the substance's assets.

It is that this SSH that SSH-Manager attempts to offer a more secure manner to get admission to the far-flung servers. SSH authenticates a consumer the use of a personal key this is saved mystery through the consumer. This personal secret of maximum significance because it gives complete get admission to a far-off server. Managing gets entry to the servers has been a fantastic concern and hassle in the maximum of the organizations.

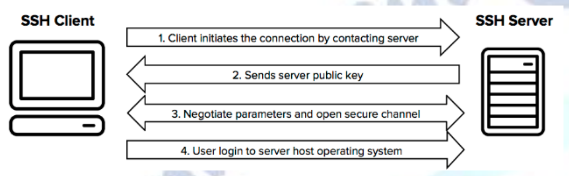

SSH Client
SSH Server
1. Client initiates the connection by contacting server
2. Sends server public key
3. Negotiate parameters and open secure channel
4. User login to server host operating system

Figure1
Figure guide:
(a): The SSH Protocol

II. LITERATURE SURVEY

A- Related Applications

Automation of SSH key management [1]

Management of user keys for public key authentication using the SSH in large SSH deployments is automated by deploying a management system in the environment, discovering SSH identity keys and authorized keys, analysing authorized connections between user accounts, and automatically managing the authorized connections and the key pairs used for authentication.

Setting up centralized Certificate Authorities for generation of public keys [2]

The main purpose of a Certificate Authority is to sign a key. We can determine from a public key, whether it is signed by a legitimate Certificate Authority or not. Since, signing a key is a crucial process of a system, hence the Certificate Authority should be centralized, meaning, it should be hosted on an instance managed by a single entity. This helps in preventing any sort of malpractice which may occur in case of a distributed Certificate Authority.

Using Secure shell for securing Hadoop Clusters [3]

Secure shell or SSH is just a way of authentication (just like usernames and passwords). Hence, it can be used for authorization in almost all of the scenarios. One such scenario is securing Hadoop clusters. Hadoop is a framework which allows processing of large-scale data and a cluster represents a group of computers. Hence, we can secure all the machines in this clusters using SSH protocol. The main benefit of using SSH is that it is resistant to brute force attacks, and all the traffic that travels to and from the servers remains encrypted.

Good practices for managing SSH keys and challenges faced [4]

US National Institute of Standards and Technology (NIST) issued guidance on SSH key management as NIST IR 7966. It is a good starting point for understanding how to manage access using SSH. We wrote most of the NIST guidelines, and have expanded upon them in our internal processes. We also invented SSH (Secure Shell). We are the best subject matter experts in the field.

A system for easing the pain of managing key / user / account associations [5]

Managing SSH keys can become quite a pain when working with large SSH deployments. Hence a centralized system is required for automatically managing SSH keys and granting and revoking access to particular servers.

Use of two-factor authentication for remote logins. Identification and management of remote users [6]

Apart from using SSH keys (which in itself is quite sufficient), we can make the authorization process more robust by introducing two-factor authentication. The process of two-factor authentication includes an extra step in the process of use authorization. The additional step is usually an OTP via an authenticator application (Google Authenticator, Microsoft key generator etc).

Cryptographic key management challenges in the context of architectural solutions [7]

Apart from the problem of managing SSH keys, in a large deployment system, we can encounter the problem of adding the SSH key management system in the current infrastructure. Hence, key changes in infrastructure might have to be made in order to make this system feasible.

III. METHODOLOGY

In this paper, we present you an automated SSH key management system which will have some keys features such as: Automated access control, Automated system deployment, Temporary key generation, Automated key deployment, Robust access control and top of the line cryptographic security.

Architecture Diagram

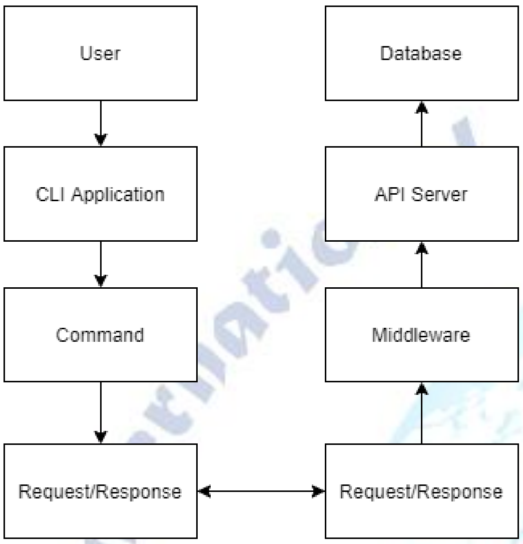


Figure 2
Figure guide:
(a): Overview of Architecture

Implementing Authentication

Authentication was implemented from the basics. All the concepts were implemented from scratch.

Signup: During signup the CLI sends registration info to the backend and the backend then creates multiple things for creation of user account. The backend creates a random salt, an encryption/decryption key, a password based key encryption/decryption key, initialization vectors and hash of the password received. The encryption/decryption is used to encrypt the SSH keys of the user and it is stored in DynamoDB after being encrypted with the help of the password-based encryption/decryption key which can only be generated using the user password. Everything generated at this stage except password-based encryption/decryption key is stored in DynamoDB.

Login: For login the username and password are sent to the backend and a hash is calculated using the password and salt of the user. If the hash generated matches with the hash stored in the DynamoDB then the user is verified and an access token is returned to the CLI for storing and authentication with further requests.

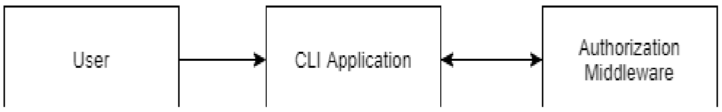


Figure 3
Figure guide:
(a): Authorization Flow

User Secret Generation

This is a very critical part of this application. During the process all the secrets of the user are generated. These broadly include the following:

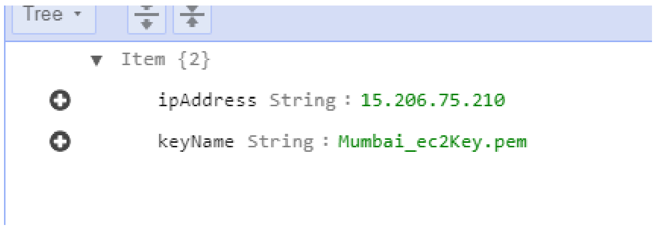
- Salts
- Initialization Vectors (IV)
- Decryption key

The decryption key is encrypted using the user's password using "Password based key derivation function 2 (PBKDF2)". The detailed list of all secrets generated are given in the table below.

Name	Size (Bytes)
Decryption Key (dek)	32
IV for dek	16
IV for Key encryption key (kek)	16
Salt for dek	32
Salt for kek	32
Salt of password	32

Creating Instance Mapping

Once the user has created an account and successfully logged in, then the user can create instance mapping with the help of a single command on the CLI.
After the command is executed on CLI, the backend fetches list of all the available instances from all the AWS regions and creates a mapping of EC2 instance IP addresses along with their SSH keys' names.
The mapping is stored in the DynamoDB and needs to be updated if the user creates a new Amazon EC2 instance.



```
▼ Item {2}
    ipAddress String : 15.206.75.210
    keyName String : Mumbai_ec2Key.pem
```

Figure 4

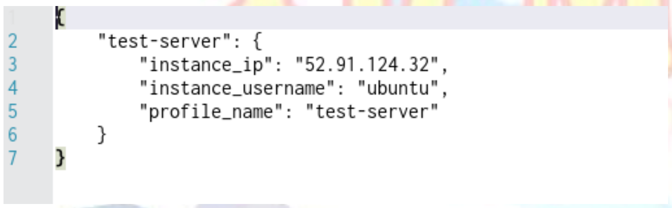
Figure 4 guide:
(a): Example of Instance Mapping

Uploading SSH Keys

After the instance mapping is created and stored the user has to upload SSH keys via the CLI app. The keys are sent over a secured channel and encrypted to the backend and then is again encrypted using the encryption/decryption key which was encrypted using the password-based encryption/decryption key.
Once the SSH key is encrypted it is then stored inside the DynamoDB.

Adding Profiles

Profiles are a neat way to create and handle SSH configurations for the user. Once a profile is created the user then needs not to provide the IP address and host of the Amazon EC2 instance every time.
User can secure shell into the EC2 machine just by referring the name of the profile and after that everything is handled by the CLI app after that.



```
{
    "test-server": {
        "instance_ip": "52.91.124.32",
        "instance_username": "ubuntu",
        "profile_name": "test-server"
    }
}
```

Figure 5
Figure 5 guide:
(a): Profile Example

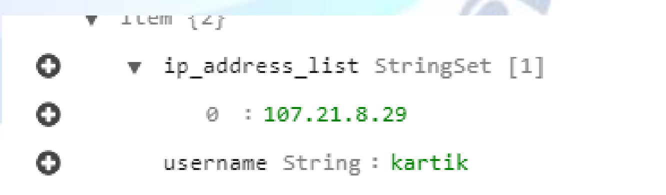
Certificate Authority

A certificate authority (CA), also sometimes referred to as a certification authority, is a company or organization that acts to validate the identities of entities (such as websites, email addresses, companies, or individual persons) and bind them to cryptographic keys through the issuance of electronic documents known as digital certificates. A digital certificate provides:
Authentication, by serving as a credential to validate the identity of the entity that it is issued to.
Encryption, for secure communication over insecure networks such as the Internet.
Integrity of documents signed with the certificate so that they cannot be altered by a third party in transit.

In In this paper, we have automated the process of setting up a certificate authority which automatically generates a digitally signed temporary SSH key for the users. We have used a tool called Ansible for automating the entire setup process.
Before generating a temporary SSH key, a trusted CA key-pair is to be generated which is kept safely. The public key of the generated key-pair is copied to the server's "/etc/ssh" directory. After that, an entry is created in "/etc/ssh/sshd_config" specifying that the copied key is a "Trusted CA key". This public key is further used for signing all the temporary key(s).

Access Control

Apart from managing the SSH keys, we have created the ability to provide server-based access to the users. With server-based access, we can restrict a user's access to a limited number of servers. This is achieved by assigning just the required servers' IP addresses to the user thereby granting him/her the access to those IP addresses. Whenever a user is given access to a server, an asymmetrically encrypted API call (consisting of the username and IP address(s)) is sent to the backend server instructing the server to grant the access of the provided servers' IP addresses to the specified user.
Similar process is followed for revoking the access but the only difference here is that before revoking access, a check is performed whether the user had the access to the server for which his/her access is being revoked. The benefit of carrying out this check is that it saves redundant database calls.

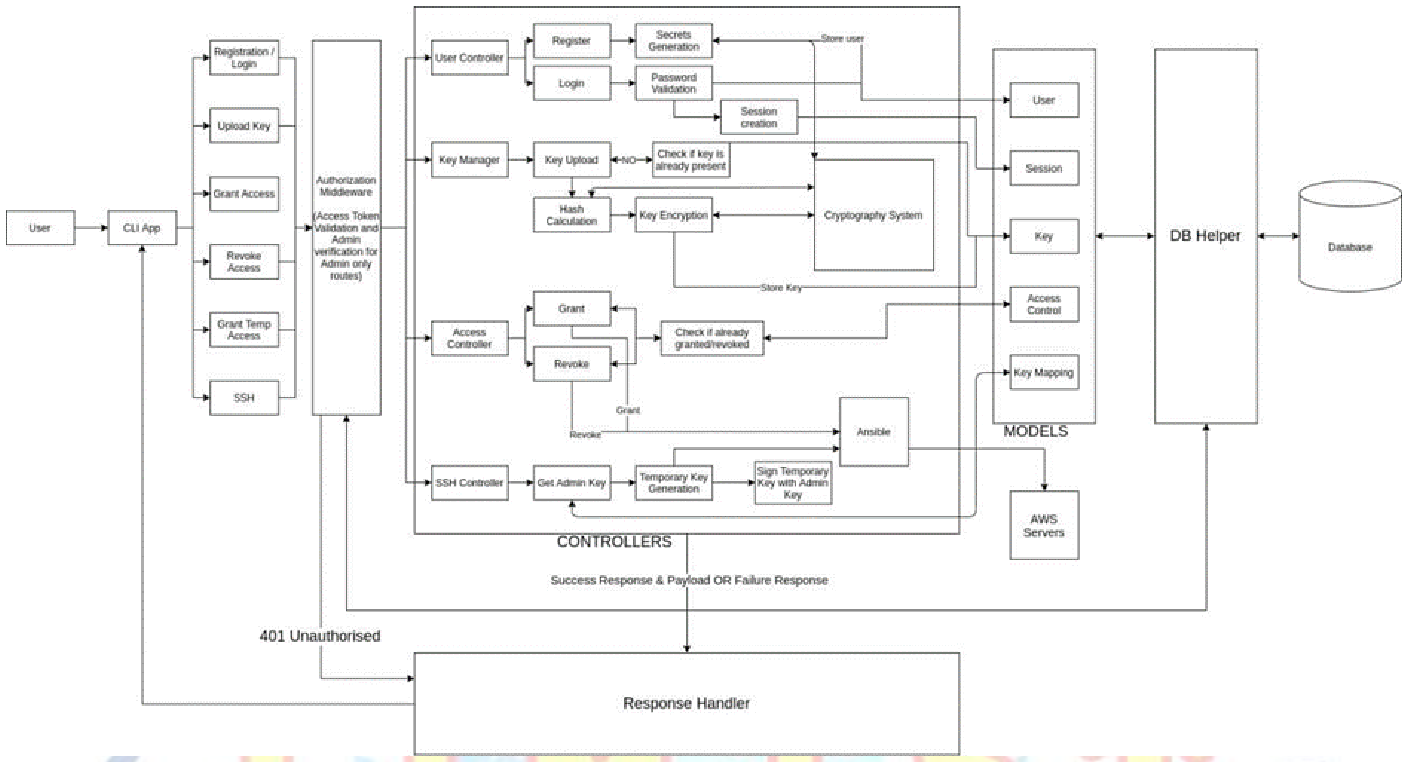


```
▼ item {2}
    ▼ ip_address_list StringSet [1]
        0 : 107.21.8.29
    username String : kartik
```

Figure 6
Figure 6 guide:
(a): Access Control Example

IV.RESULTS

In this paper we have implemented a tool which automates the process of SSH key management. The detailed architecture of the application is given below.



REFERENCES

[1] Tatu J. Ylonen, "User Key Management for the Secure Shell (SSH)", 2018.

[2] Dell –Michael, "Setting up Certificate Authority Keys with OpenSSH", 2011.

[3] Huixiang Zhou et al., "A New Solution of Data Security Accessing for Hadoop Based on CP – ABE", 2014.

[4] Kent, Greg, "Unsecured SSH —The Challenge of Managing SSH Keys and Associations", 2010.

[5] D. Arkhipkin et al., "An SSH key management system", 2008.

[6] The Start Experiment, "An SSH public key management system".

[7] Ramaswamy Chandramouli, Michaela Iorga, Santosh Chokhani, "Cryptographic Key Management Issues and Challenges in Cloud Services", 2013.