2021-06-14

Simple SSH Management

Torstein Calvin Collett
Brigham Young University

Simple SSH Management

Torstein Calvin Collett

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

ABSTRACT

Simple SSH Management

Torstein Calvin Collett
Department of Computer Science, BYU
Master of Science

SSH certificates are used by administrators so connections to the server can be verified. This ensures that only authorized administrators can access the server and that the server being accessed is the intended machine. Current solutions for managing SSH certificates are focused on commercial use, which makes them cumbersome for small groups and individuals to use. These solutions require running multiple services that companies already use but add significant overhead for smaller groups. We developed a new standalone system that makes it easy to manage SSH certificates for small amounts of servers and users, without requiring additional servers to be deployed. We evaluated our system with a user study to demonstrate its ease of use. We hope that this implementation can help guide future research toward a more simplified certificate authentication system for SSH.

Keywords: SSH, certificate-based-authentication

Table of Contents

List of Figures

Chapter 1

Introduction

Current practice for SSH authentication typically involves having a system administrator log in using either a password or their SSH key. Using passwords is not a recommended practice because attackers can easily break weak passwords, and it is difficult to get people to memorize long, random passwords. Using an SSH key involves having each user generate a public-private key pair and then securely giving the administrator their public key to be put on the remote computer, allowing access.

These standard practices lead to two primary issues: (1) maintenance of user credentials and (2) Host key authentication [9]. The current practice for managing user credentials is to create a file filled with a public key for each user that has access to the computer. Host key authentication is left to manual inspection of a certificate fingerprint by the user that logs in.

Key management for SSH login is cumbersome to maintain because it is hard to keep track of different keys and to whom they belong. This hard-to-track setup can create security problems when removing old keys because it is hard to find and identify all of them. The key file needs to be replicated and tailored for each server to contain the correct users having access to each server. With multiple servers, there is a high probability of human error while setting up and correctly maintaining user access. This method for user authentication does not scale well and is hard for administrators to monitor and maintain.

Trust on first use (TOFU) is the most common way to authenticate a server. TOFU is bad practice since it allows for potential man-in-the-middle (MITM) attacks. Even though defaulting to manual verification of server connections is more secure, users rarely check the host fingerprint to make sure they are connecting to their intended server. If the server is not verified, there is no guarantee that the server is the machine the user wanted to access.

Current[5, 7] solutions are enterprise solutions developed by and for large organizations. Enterprise solutions work well to solve both host verification and user authentication. But they rely on integrating with other enterprise software to be most effective. These solutions use Identity Access Manager (IAM) systems for authenticating users and some use proxy machines to help monitor and audit connections. Setting up an IAM account and a proxy just to run several servers creates an overly complicated setup process for a small group of users or an individual who manages a few machines.

We developed a small-scale solution that is easy to deploy. This solution addresses both management of user authentication and authentication of hosts. The primary purpose of our system is to create the bare minimum service for allowing SSH certificate authentication between client and server. This system is designed to be standalone without the need for integrating with other services and will keep the setup as streamlined as possible. The SSH certificates for clients and servers are managed together through an easy-to-use mobile application. This app gives the administrator the ability to generate certificates for clients and servers. Each server and client get its certificate from a cloud database along with the root certificate to verify connections that it makes or receives. The mobile app stores the hash of public keys in a cloud database for clients and servers to know which certificate is theirs. The mobile app will give push notifications to the administrator to remind them when user credentials need to be reviewed for renewal.

To test our system we did a user study on college students that had used SSH before. The participants interacted with both the app that we created and the setup script for setting up both server and client machines. We had our participants set up two servers, a client computer, and a client certificate. After completing the tasks, the participants filled out a survey to determine the perceived usability of our system. We then had a recorded interview to help us better understand what they thought of our system.

The results of our study indicate that our system was usable with a System Usability Scale (SUS) score of 74. Most of our users said they could see themselves using our system to manage a setup of servers. We received feedback from our participants for improving our system to make it better and found other ways to improve it while observing the participants complete the tasks given.

Chapter 2

Related Work

Tatu Ylonen has summarized two issues with the current usage of SSH keys[9]. These two issues are, trust on first use (TOFU) for host authentication and management of user authentication keys. In his paper, he discusses a variety of solutions, why they are insufficient, and lists characteristics that a great solution needs to have. The desired attributes of user authentication are: ensure that access is terminated when the user leaves, implement and enforce proper approvals and accountability for machine-to-machine authentication, access to servers should be fully and easily auditable, it should be possible to delegate who can grant access to which accounts, rotating user credentials should be supported, use of access credentials should be monitored, and it should be possible to centrally identify and revoke access rights. The desired attributes for server authentication are: host authentication checking should be automatic and enforced by default, initial deployment should be easy, server upgrades and host authentication rotation should be automatically supported, and it should work for clients behind a NAT. General requirements for a new system described by Tatu Ylonen are: able to support legacy software, able to work across organizational boundaries, no single point of failure, no denial of service opportunities, and should not add excessive latency.

Ed Dawson et al [4] wrote an overview of the best ways to generate, distribute, and store keys used for authentication. The best way to store keys is to have some specialized hardware to keep the keys secure. This paper gives a good overview of how to keep keys safe and secure which translates into secure practices for the certificates for our system.

D Arkhipkin et al [2] built a system in 2008 to help with key management in the STAR environment. The system allows for tracking of where SSH keys are used to determine if a key was leaked. They also created a web interface that helps administrators keep track of who has access to their servers.

Several previous works created different solutions for verifying the identity of servers. J. Schlyter and W. Griffin [6] created a proposal to use the DNS for verifying SSH host machines with security features from Domain Name System Security (DNSSEC). Since DNSSEC is currently not widely used, we will not use it in our architecture, but it could be once DNSSEC is more widely adopted. Yasir Ali and Sean Smith [1] created a database to store their public keys in using passwords to generate a MAC to preserve the integrity of the public keys. This method allows for verifying the server from any device with their software installed even when the server is getting connected to for the first time. Dan Wendlandt et al. [8] talk about using a notary system to prevent the wrong certificates from servers from being passed to the client. The main idea behind their work is to make multiple requests to different machines for the same information. If someone is trying to MITM their connection, they would not be able to intercept all the unique connections to the notary servers and the main website's server. They also talk about notaries auditing each other to prevent malicious notaries from popping up and ruining their verification system.

Recent work has focused on managing certificates for servers and users instead of keys. Certificates have the advantage of not needing a list of public keys stored on each server and client. Servers only need to save the public keys of the authorized certificate authorities. Certificates also allow for more control over the time a user can log in to a server through expiration dates. A single certificate can give authorized access to multiple computers.

Current systems that use certificates were built for large enterprises. Three main company solutions are out on the market. These solutions are Teleport[1] from Gravitational,

[1]https://gravitational.com/teleport/

PrivX[2] from SSH, and Smallstep[3] from Smallstep. All of the programs discussed above focus on integrating with identity and access management programs commonly used when large groups need access to servers of different varieties. All of these solutions address both TOFU and user authentication with infrastructure that gives certificates.

Smallstep requires three types of services running. These services are a certificate authority (CA), Identity Access Manager (IAM), and SSH on the host computers. Smallstep works by having the CA set up with a list of authorities that can verify the identity of different computers and users. When using Smallstep with Amazon, the Smallstep CA can determine which host it should issue certificates to depending on the Amazon accounts linked to the CA. The CA will only verify AWS instances that are within the linked Amazon accounts. The main focus of Smallstep is to implement machine-to-machine authentication. Adding multiple authentication services allows integration of IAM and any other authentication service that uses JSON web tags (JWT).

PrivX and Teleport are both based on the same architecture pioneered by Netflix in their system called BLESS (Bastion's Lambda Ephemeral SSH Service) [5]. These implementations integrate with IAM systems that are used by companies for authentication. The use of IAM allows employees to use one central company account to authenticate. The program gives employees access to the servers they need by issuing short-lived certificates. They use short-lived certificates to log in users to servers for one-time use access. PrivX and Teleport use a proxy server with web-based access that allows client computers to use any browser. Through the proxy server, all SSH sessions get recorded for auditing purposes. The servers are all set up to only accept connections from a group of proxies that users access using the IAM system. They can then ssh into any computer that their user group is allowed to access through the proxy. Using short-lived certificates and authorized IP addresses helps to keep client authentication monitored for companies.

[2]https://www.ssh.com/products/privx/
[3]https://smallstep.com/

Chapter 3

System Design

Our system focuses on minimizing the number of additional servers that the administrator needs to deploy to manage access to servers. Figure 3.1 shows how we designed our system. The administrator uses a smartphone application that creates certificates, then places them into Firebase. These Certificates are then fetched from Firebase by the clients and servers. After fetching their respective certificates, and the root public key, the clients and servers can verify each credential by relying on the root public key of the phone only signing valid certificates. By fetching the root public key, servers and clients connect without relying on TOFU.

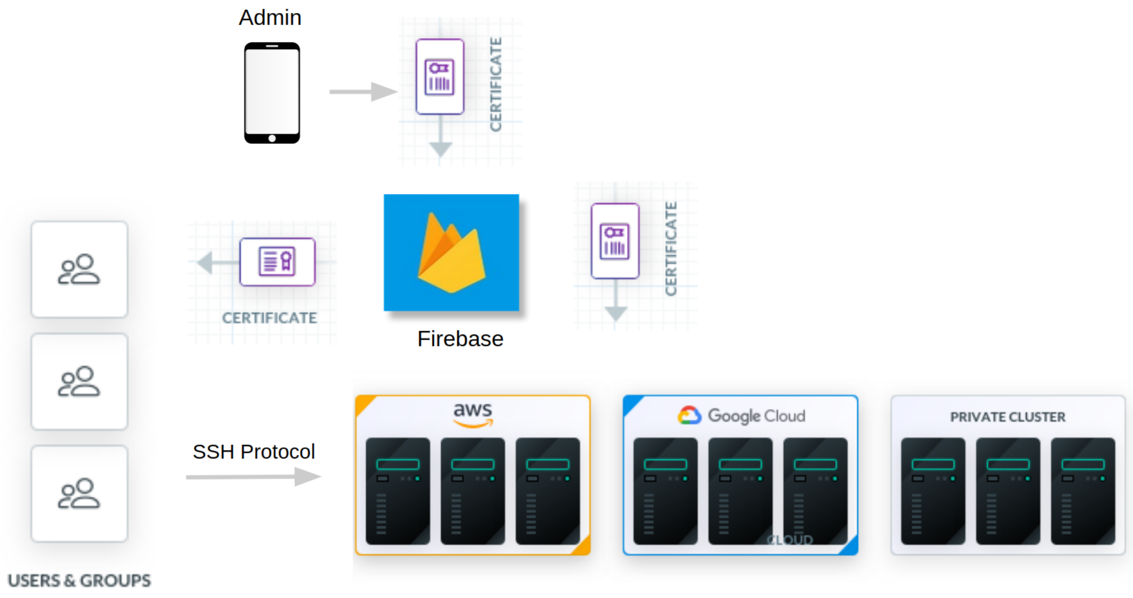


Figure 3.1: The finalized design of our prototype SSH software. Certificates are generated on the phone and distributed to the proper machines through Firebase.

One of the purposes of the smartphone app is to be the certificate authority for both users and servers. The other purpose of the app is it allows the administrator to have quick and easy access to see which certificates need renewal.

We designed our app using the Flutter UI toolkit. Flutter allowed for the quick creation of our app for both Android and iOS. While developing our app, we could not find a suitable Flutter package for generating ssh-keys and Openssh certificates. Because of limited development time, we opted to deploy a local certificate authority on an additional server, which uses the terminal command ssh-keygen to generate our certificate rather than building a Flutter package for generating SSH keys and Openssh certificates. For the user study, we ran our certificate authority locally on a lab computer.

Along with the smartphone app, we made a simple bash script that would help automate the setup for client and server machines. This script makes configuration changes to SSH that allows certificate verification and authentication through principals. The setup scripts we created enable the user to fetch a new certificate and have a cron job that checks for a new certificate daily.

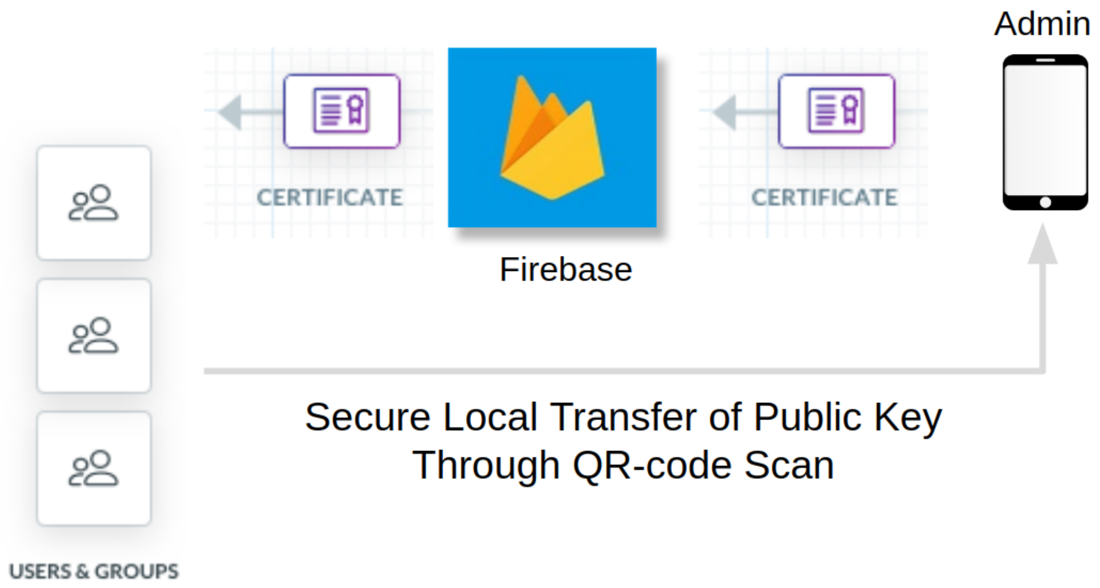


Figure 3.2: Architecture for the generation of certificates and their propagation to the proper server or clients.

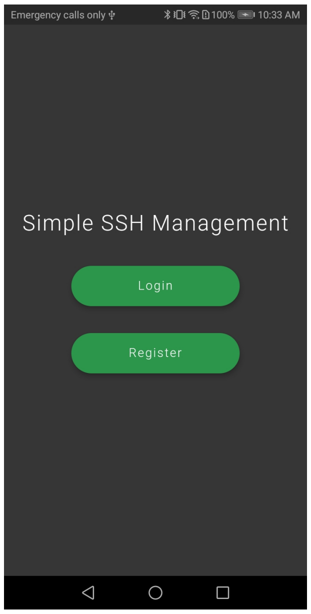
Figure 3.2 shows the process for generating certificates. The setup script is run on the server or client machine to set up the configuration files. While this script is running, it prints out a QR code containing the public key that needs to be signed to become a certificate. The administrator scans the generated QR code with the app. Then sets the properties for the certificate getting generated. After the generated certificate on the phone gets saved into Firebase, the client machine or server can then fetch the certificate from Firebase using a SHA-256 hash of the public key and the admin ID corresponding to the app account.

For server computers, the admin starts by inputting the users allowed to accept SSH connections through certificates. After inputting the first username, the script then asks for the issued principals authorized to log in as that user. After inputting the principals, the script loops back, asking for another user to set up and then asks for the new user principals. After setting up all the users, the script then generates a public-private key pair using ssh-keygen. The public key is then displayed as a QR code in the terminal using UTF-8 art. The terminal window then directs the admin to scan the QR code with the smartphone app. Once the certificate gets uploaded to Firebase, the script then tries to fetch the signed certificate by prompting the admin to type in the admin ID. The admin ID indexes into the database, so the computer gets the certificate from the correct admin account.

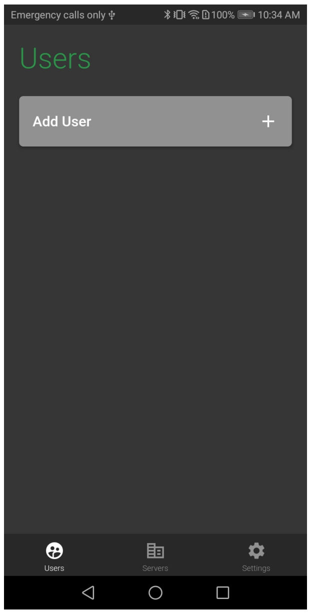
Figure 3.3 is a collection of screenshots from the app. The following paragraphs will describe each of the screenshots in-depth.

Figure 3.3a is the first screen that a user sees when opening the app for the first time. If they have already logged into an account, the app immediately goes to the login screen. The login screen will then only allow them to log in as the previously logged-in account to prevent the use of their private key on a different account.

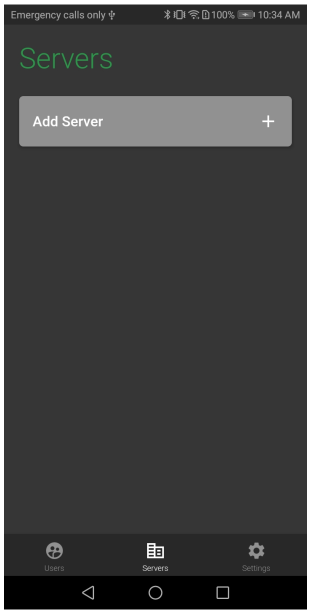
Figure 3.3b shows the tab that lists all user certificates generated in the account. The generated certificates can be modified or renewed by pressing the edit button next to each certificate.



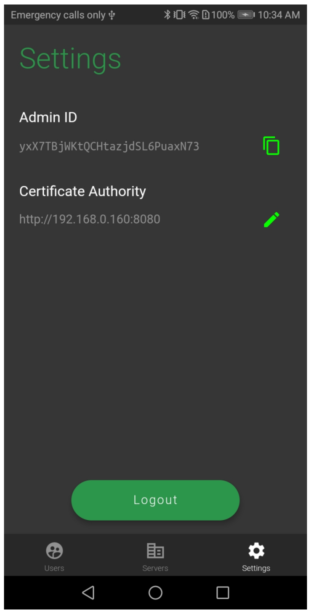
(a) Opening page where users can login or register a new account



(b) Tab to display all users with a certificate



(c) Tab to display all the servers with a certificate



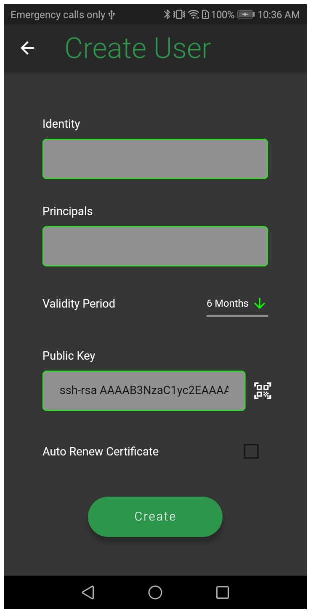
Admin ID

yxX7TBjWKtQCHtazjdSL6PuaxN73

Certificate Authority

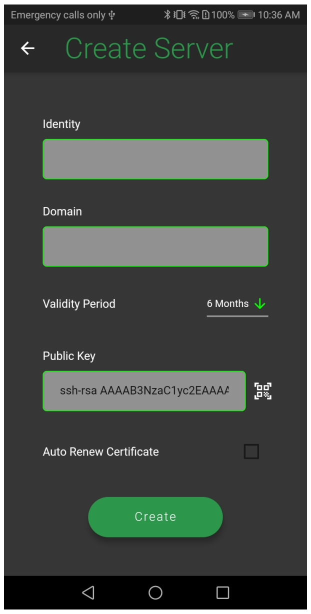
http://192.168.0.160:8080

(d) Settings tab which displays the admin ID to get certificates from Firebase



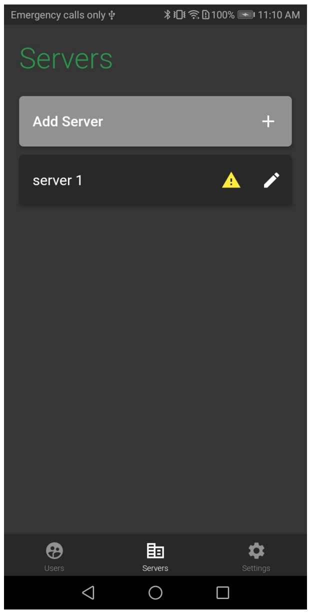
ssh-rsa AAAAB3NzaC1yc2EAAAA

(e) Page for creating a user certificate with public key scanned

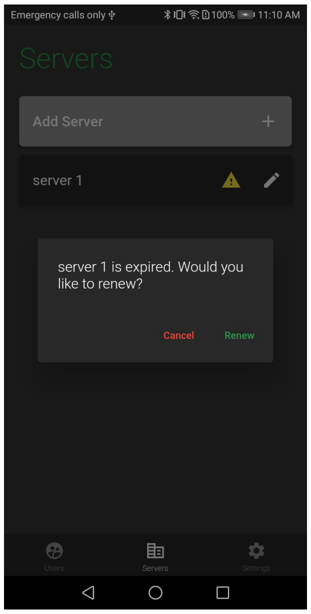


ssh-rsa AAAAB3NzaC1yc2EAAAA

(f) App page for creating the server certificate with public key scanned



(g) Example of Server screen with an invalid certificate



server 1 is expired. Would you like to renew?

(h) Popup window that allows for quick renewal of a certificate

Figure 3.3: Screenshots of different screens in the app used to setup certificates for both servers and users

Figure 3.3c shows the tab that lists all server certificates. This tab has the same functionality as the user certificates, except that these certificates are SSH host certificates.

Figure 3.3d shows the settings tab that currently displays the admin ID for the account and the location on the network to find the python certificate authority that we used for the study.

Figure 3.3e shows the create user screen that allows input for specifying what goes into the user certificate. There is a drop-down for a variety of times for the validity period of the certificate. Instead of needing to type in or copy-paste the public key, the app scans a QR code of the public key generated by the setup script.

Figure 3.3f shows the create server screen used to specify what is in the server certificate.

Figure 3.3g shows the server screen with a certificate that has expired. This icon allows the user to know, in the app, which certificates have expired. Pressing on the yellow triangle brings up the popup window, shown in figure 3.3h, allowing for a quick renewal of that certificate. The app also allows for resigning of certificates with the same settings. Early re-signing is done by pressing the save button on the edit page without changing any fields.

Along with the features described above, our app also could notify the admin when a certificate expired. However, we did not use this functionality during the user study.

Chapter 4

Methodology

4.1 User Study

Brigham Young University Institutional Review Board approved our user study. In this section, we discuss how our study was conducted.

4.1.1 Study Setup

The study ran for just under three weeks — beginning Friday, April 2, 2021, and ending Thursday, April 22, 2021. Twenty participants completed the study. Participants took between 20 and 30 minutes to complete our user study and received compensation of 10 USD.

4.1.2 Demographics

We recruited participants from BYU's campus that were familiar with using SSH. We targeted our advertising toward computer science, information technology, and engineering students. We put up flyers in each of the colleges' respective buildings and advertised the study in several computer science classes and the computer science newsletter. We included the flyer in appendix A.

All our participants were male. Participants ranged in age between 20 and 33 years old, with 15 (75%) of the participants under 25. All participants were current students at BYU.

4.1.3 Scenario Design

The participants were asked to role-play as a system administrator setting up new servers and giving employees access. They were shown the following text.

```
Imagine that you are a system administrator and needed to start using
our system to give developers access to different servers. Our
system is designed to gather authorization information into an app
on your phone to allow quick and easy access to check on who has
access to the systems you manage.
```

Participants were walked through each step and were given passwords to use when passwords were needed.

4.1.4 Task Design

We conducted a user study with several different tasks to help our participants get a feel for using the app to evaluate our new system. The tasks in the user study were:

- Set up two different servers with different principals.
- Be the system administrator and generate a certificate for the study coordinator that expires after two minutes.
- Be an employee who gets access from the study coordinator (acting as the system administrator) to only one of the servers previously set up.

The participant task guide is in appendix B.

4.1.5 Study Questionnaire

After a participant finished with these tasks, we gave them a survey to complete. This survey gathered a System Usability Scale (SUS) score, their impressions of what they thought of the system we built, and some demographics. We also collected how data on how many servers they currently manage. We included our survey questions in appendix C.

4.1.6 Post-Study Interview

Along with filling out the survey, we asked the participants questions verbally about how they felt about our system and recorded their answers. We had them explain their general experience with the SSH management system and anything they would like to have different. We asked them what they liked and did not like about their experience using our system. Along with questions about the user experience, we asked the participants if they would use it in any servers they are currently managing or in specific use cases. Most participants split the use case between home use and work use.

4.1.7 Limitations

We focused our study on a sample of students that would potentially be going into jobs where they would manage groups of servers. We did this in hopes of getting a sample that represents future users of our type of system rather than those who wouldn't need it. Because we focused on students at BYU, our study is not representative of all system administrators who might consider using our system.

Chapter 5

Results

5.1 Perceived Usability (SUS)

We evaluated our system using the System Usability Scale[3] (SUS) questions in our survey to obtain a measure of its perceived usability. SUS is measured by asking a set of ten questions on a scale of one to five and then calculating a score out of one hundred. Each question has a final weight of ten points. The final score gets categorized into letter grades and adjective ratings. The categories get split as follows: 80.3% or greater is considered an A (Excellent), 68% to 80.3% is B (Good), 68% is C (Okay), 51% to 68% is D (Poor), and less than 51% is F (Awful). The average SUS score from our twenty participants was 74%, with the lowest being 55% highest of 83%, with a median of 75%. Having an average SUS score of 74% makes our program fall into the "Good" classification with a letter score of B.

5.2 Intention to Use

In the survey, we asked the question "If I needed to manage a small number of servers (less than 10) in the future, I would like to use this system." The majority of our participants responded to this question as Agree (12; 60%), The next highest was Strongly agree (6; 30%), with the last two responses being Neither disagree nor agree (1; 5%) and disagree (1; 5%). While reviewing the audio interviews, we noticed that there were varying ideas on how big a server setup our users would want to use this system. Some participants said they would use it for only a single server, while others felt the system would be useless unless they managed more than a few servers. One participant said, "I would like to use it. I have one server that just runs at my house that occasionally I need to give ssh access to people." Another participant said, "I think if I only had 1 or 2 servers, I might not use it because I might not need an app like this but, I feel like if I had more than that, I would definitely see myself using it."

5.3 Technical Problems

We had a couple of technical problems with the user study setup, but it only affected two of our participants. The first issue that occurred was that the phone app was freezing during the sign-in process. We were able to figure it out rather quickly, learning that the phone was not finding the certificate authority on the network because it connected to the wrong Wi-Fi network. The second bug was in our code. For some reason, the app makes duplicate entries for a certificate on Firebase if you use the default timeout option. When asked in the survey if they had technical problems, the majority said "No Problems" (16; 80%), with the rest saying "Few Problems" (3; 15%), and "Some Problems" (1; 5%). Most technical problems mentioned in the interview question were from typing in the admin ID incorrectly. Some of the participants reported confusion when setting up the user on the computer. This confusion stemmed from the script using a loop to allow inputting multiple users, and they only had to set up one user.

5.4 Things Participants Liked

When we asked about what participants liked, most of them mentioned using the QR code to transfer the public key as a cool feature. They also liked being able to manage server access in one centralized location. Along with having one centralized location they liked the ability to do it from anywhere on their smartphone. The most common comments given were that the system was easy to use and it was quick.

5.5 Things Participants Disliked

The most disliked part of the entire experience, according to our participant interviews, was needing to type in the admin ID. When asked about what they disliked, one participant said, "I think the biggest thing was just the long ID that I had to type in." The majority of our participants echoed a similar sentiment about the admin ID. Some participants felt it had too much overhead from the setup if only setting up one server with only one user. One of our participants disliked automation and said he would prefer to do all the set-up manually, so he knows exactly what is going on.

5.6 Discussion

The results of our study indicate an overall positive response to our system. A SUS score of 74 shows that our system is usable and can benefit those who manage systems. The study also indicated our system has room for improvement that would help make it flow better and be easier to use.

Several features would allow our Simple SSH Management system to become more robust. The main improvement would be implementing ssh-keygen inside the app to making all key signing occur on the phone, as the original system design specifies. Other features include adding the ability to revoke certificates and have the ability to rotate keys automatically. One improvement for the terminal scripts would be to package it together and set it up as a terminal command with a man page telling how to use all of the features built-in rather than just the setup script. Another improvement would be to shorten the admin ID. A shorter admin ID would allow for fewer user errors and would be easier for users to type.

Chapter 6

Conclusion

We built a simple SSH management system that helps the average user of SSH start authenticating with certificates. Our system removes the need for TOFU and lowers the cost of maintaining correct access privileges on each server. The user study we conducted shows that our system architecture is a viable solution for small setups to minimize deployed servers and still benefits from certificate authentication.

This system is applied best in settings that want to minimize setup overhead and potentially have high user turnover rates. Some use cases include giving students short-term access to school research lab servers, smaller start-up companies that use contract workers or high employee turnover, or securing ssh access to hobby servers.

The prototype we developed is currently not production-ready as it still needs to implement ssh key signing natively in the app rather than relying on the python CA used in the study. Our solution makes certificate authentication more accessible to those who manage smaller infrastructure and don't want to host a setup with an IAM service.

References

[1] Yasir Ali and Sean Smith. Flexible and scalable public key security for SSH. In *European Public Key Infrastructure Workshop*, 2004.

[2] D Arkhipkin, W Betts, J Lauret, and A Shiryaev. An SSH key management system: easing the pain of managing key/user/account associations. *Journal of Physics: Conference Series*, 119(7):072005, Jul 2008. doi: 10.1088/1742-6596/119/7/072005. URL `https://doi.org/10.1088%2F1742-6596%2F119%2F7%2F072005`.

[3] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Studies*, 4(3):114–123, May 2009. ISSN 1931-3357.

[4] Ed Dawson, Andrew Clark, and Mark Looi. Key management in a non-trusted distributed environment. *Future Generation Computer Systems*, 16(4):319 – 329, 2000. ISSN 0167-739X. doi: https://doi.org/10.1016/S0167-739X(99)00056-4. URL `http://www.sciencedirect.com/science/article/pii/S0167739X99000564`.

[5] Russell Lewis. How Neflix gives all its engineers SSH access to instances running in production. `https://www.oreilly.com/library/view/the-information-security/9781491968345/video248959.html`, 2016. Presentation at Oscon.

[6] Jakob Schlyter and Wesley Griffin. Using DNS to securely publish secure shell (SSH) key fingerprints. *RFC4255, Jan*, 2006.

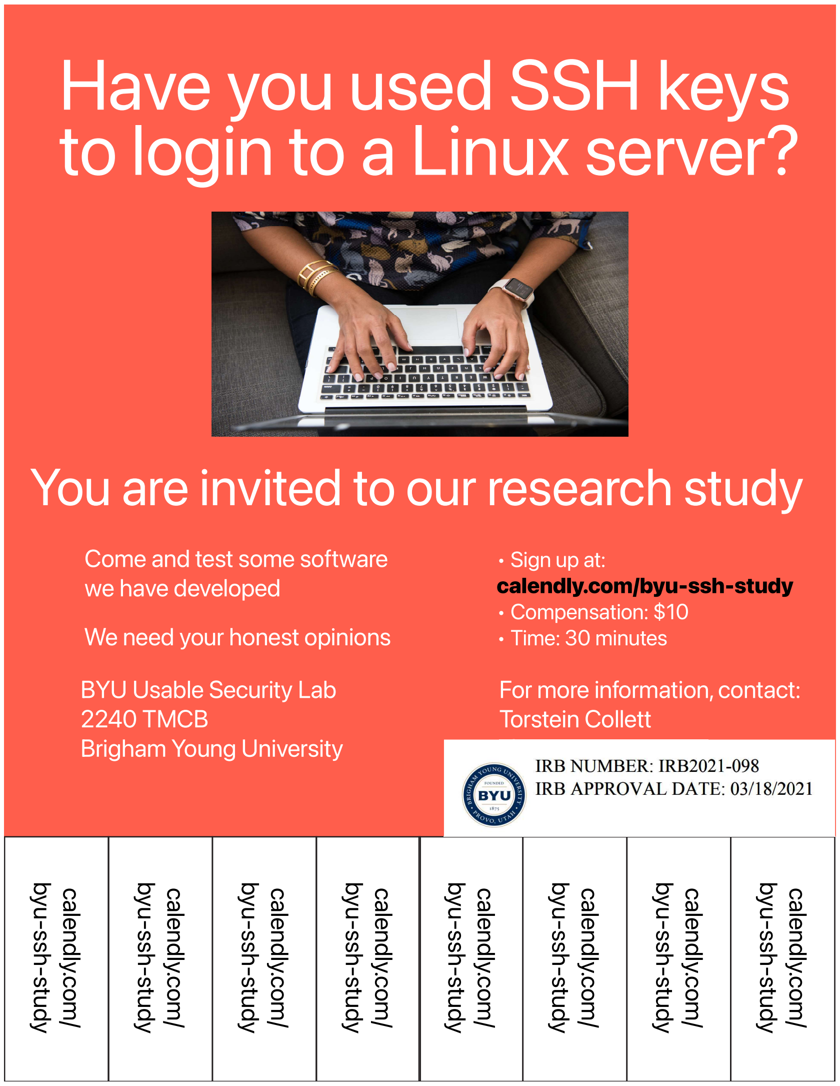
[7] VA Stafford. Zero trust architecture. *NIST Special Publication*, 800:207, 2020.

[8] Dan Wendlandt, David G Andersen, and Adrian Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX Annual Technical Conference*, volume 8, pages 321–334, 2008.

[9] Tatu Ylonen. SSH key management challenges and requirements. In *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2019.

Appendix A

Recruiting Flyer



Have you used SSH keys to login to a Linux server?

You are invited to our research study

Come and test some software we have developed

We need your honest opinions

BYU Usable Security Lab
2240 TMCB
Brigham Young University

• Sign up at:
calendly.com/byu-ssh-study
• Compensation: $10
• Time: 30 minutes

For more information, contact:
Torstein Collett

IRB NUMBER: IRB2021-098
IRB APPROVAL DATE: 03/18/2021

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

calendly.com/ byu-ssh-study

Figure A.1: Flyer put up in the different college buildings

Appendix B

User Study

B.1 Participant Tasks

We provide participants with a smartphone with our Simple SSH Management app installed and a computer to use for accessing servers via SSH.

Imagine that you are a system administrator and needed to start using our system to give developers access to different servers. Our system is designed to gather authorization information into an app on your phone to allow quick and easy access to check on who has access to the systems you manage.

B.2 App Setup

Use the provided smartphone to create a new account in the Simple SSH Management app.

1. Enter email (marcopolo@gmail.com)
2. Enter password (marcopolo42)

B.3 Server Management

You will play the role of a manager setting up two servers and giving one employee access for a brief amount of time.

1. Use SSH to connect to the server www.marco.com using the username: sysadmin and password: sysadmin42.
   - (ssh sysadmin@www.marco.com)
2. Change into the directory where the Simple SSH Management server software is located.
   - (cd SSHManagementApp/server_client_install)
3. Run sudo ./setup.sh to start initializing the server.

(a) Type in server to follow the server setup protocol

(b) When prompted allow the user test to ssh using certificates

(c) Add the password test to the test account

(d) One line at a time, add the following three principals to the account (admin dev qa)

(e) Press enter on empty lines to continue the setup process (2 times one for principals one for users)

(f) Press enter to not create a password for the private key

(g) When the QR code shows in the terminal, press the add server button from the Server List page in the phone app and scan the QR code.

(h) Set certificate properties on the app and save the certificate

- Identity: Marco
- Validity: 1 week
- Domain: (leave blank)

(i) When prompted add the account ID for downloading certificates, which can be found in the settings tab in the phone app. Make sure to double-check your input is correct.

(j) Press enter in the terminal to continue the setup script

(k) Type exit to leave the server's terminal

4. Set up a second server www.polo.com by using ssh with username sysadmin and password sysadmin42.

   - (ssh sysadmin@www.polo.com)

5. Change into the directory where the Simple SSH Management server software is located.

   - (cd SSHManagementApp/server_client_install)

6. Run sudo ./setup.sh to start initializing the server.

   (a) Type in server to follow the server setup protocol

   (b) When prompted allow the user test to ssh using certificates

   (c) Add the password test to the test account

   (d) One line at a time, add two principals to the account (admin, dev)

   (e) Press enter on empty lines to continue the setup process (2x one for principals one for users)

(f) Press enter to not create a password for the private key

(g) When the QR code shows in the terminal, press the add server button from the Server List page in the phone app and scan the QR code.

(h) Set certificate properties on the app and save the certificate

  i. Identity: Polo

  ii. Validity: 1 week

  iii. Domain: (leave blank)

(i) When prompted add the account ID for downloading certificates, which can be found in the settings tab in the phone app

(j) Press enter in the terminal to continue the setup script

(k) Type exit to leave the server's terminal

You have now finished setting up two servers that will allow certificate authenticated access to the account named test.

B.3.1 Employee Management

This exercise is to demonstrate limited-time authentication access for employees that might be contract employees that have a set end date for employment.

1. The study coordinator will play the role of an employee.
2. The study coordinator will show you a QR representation of their public key.
3. Using the smartphone app, navigate to the User list page and press the add user button, then scan the QR code.
4. Set certificate properties using the app
   - Identity: Contractor
   - Principles: dev
   - Valid period: 2 min
5. Watch the study coordinator login to the test account on the www.marco.com server using the provided certificate
6. After 2 minutes have passed, watch the study coordinator be unable to login to the server because their certificate has expired.

B.4 Client Setup

This exercise is to demonstrate how to set up a client machine on a work computer for an employee.

1. The study coordinator will play the role of an employer.
2. Give the phone to the study coordinator.
3. SSH into the client machine www.client.net using the username client and password client42.
   - (ssh client@www.client.net)
4. Navigate to the directory where the Simple SSH Management client software is located.
   - (cd SSHManagementApp/server_client_install)
5. Run sudo ./setup.sh to start initializing the client.
   (a) Type in client to start the client protocol
   (b) When the QR code shows in the terminal, show this to the study coordinator.
   (c) When prompted ask the study coordinator to give you the Admin ID for downloading certificates.
   (d) The study coordinator will authorize you to login to a server using the qa role
   (e) Press enter in the terminal to continue the setup script after the certificate has been issued
6. Use SSH to login to the 1st server using the provided certificate
   - (ssh test@www.marco.com)
7. Attempt to use SSH to login to the 2nd server and it will be denied because you, in the qa role, are not authorized to access the second server.
   - (ssh test@www.polo.com)

Appendix C

Survey Questions

System Usability Scale (SUS): Please state your level of agreement or disagreement for the following statements based on your experience with the certificate management system. There are no right or wrong answers. (Strongly disagree; Disagree; Neither disagree nor agree; Agree; Strongly agree.)

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very awkward to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Question about intended use:
If I needed to manage a small number of servers (less than 10) in the future, I would like to use this system.
(Strongly disagree; Disagree; Neither disagree nor agree; Agree; Strongly agree.) I currently manage servers
(None, 1, 2 to 5, 6 to 10, more than 10)

Technical Problems:
Were there any technical problems while using the certificate management system?
(i) No problems (ii) Few problems (iii) Some problems (iv) Many problems

Demographic Questions:

Please indicate your gender.

(i) Male, (ii) Female, (iii) Other, (iv) No answer

Please indicate your highest educational degree.

(i) High school graduate, (ii) Bachelor's degree, (iii) Master's degree, (iv) Diploma, (v) Ph.D, (vi) Other

How old (in years) are you?

Free response Please indicate if you have a computer science background.

(i) Yes, (ii) No

Please indicate your area of studies/area of work.

Free response

Appendix D

Interview Questions

1. You indicated you experienced some technical problems with the system. Please describe them.
2. How would you describe your general experience with the certificate management system?
3. What do you like about the certificate management system?
4. What do you dislike about the certificate management system?
5. Would you use the certificate management system yourself? If you would, why and what systems would you use it? If you wouldn't, why not?