Cloud Server Security

Project 1

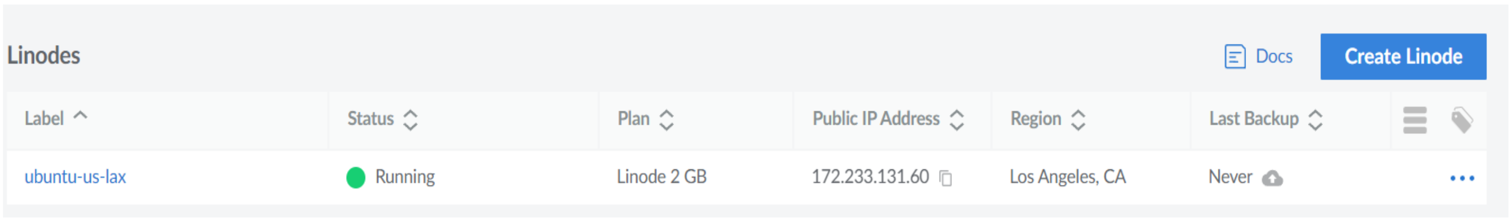
Lia Potikyan, Ara Parsyan

Basic cloud server security management, use Linux at an administrator level to properly set up all the necessary permissions, make each user responsible for the role they have been assigned, be sure to secure and defend against potential attacks such as brute forcing an SSH server, manually manage HTML based website on your server.

Building a cloud server:

A cloud server *is a virtualized computing resource provided by a cloud service provider. It operates over the internet, allowing users to access and utilize computing power, storage, and memory remotely. This model eliminates the need for physical infrastructure management and offers scalability, flexibility, and accessibility for various computing applications.*

We used Akamai Cloud manager to build an Ubuntu Server:



Label	Status	Plan	Public IP Address	Region	Last Backup
ubuntu-us-lax	● Running	Linode 2 GB	172.233.131.60	Los Angeles, CA	Never

Disabling password authentication for SSH and relying on key-based authentication:

SSH *(Secure Shell) is a secure protocol for accessing and managing remote systems. It encrypts data during communication, allowing users to securely log in, execute co mmands, and transfer files between computers over an unsecured network.*

Password authentication *is a common method for verifying the identity of a user attempting to access a system. Users enter a password associated with their account, and the system checks if it matches the stored password.*

Key authentication*, often used with SSH, employs public key cryptography for secure access. Users generate a pair of keys – a public key (shared with the server) and a private key (kept confidential). When attempting to connect, the user presents the public key. If the server finds a match with the associated private key, access is granted without a password.*

To enhance the security of our SSH access, we have transitioned from password authentication to key-based authentication on our Kali Linux system. Initially, we generated a secure RSA key pair with a bit size of 4096 using the following command:

```
ssh-keygen -t rsa -b 4096
```

the generated public key was securely copied to our cloud server, and the necessary permissions were adjusted to ensure a secure setup.

Disabling root login and user password authorization:

We modified the SSH configuration file on the server:

```
sudo nano /etc/ssh/sshd_config
```

and disabled root login and user password authorization:

```
PermitRootLogin no
PasswordAuthentication no
```

After that restart is required for the changes to be effective:

```
sudo service ssh restart
```

Installing Firewall and disabling unimportant ports:

A firewall *is a security infrastructure, either hardware or software-based, that establishes a protective barrier between a trusted internal network and external networks, such as the internet. It monitors and regulates network traffic, enforcing predetermined rules to permit or block data packets based on security criteria. By filtering and controlling data flow, firewalls help prevent unauthorized access, enhance network security, and* *mitigate potential threats, contributing to the overall integrity and confidentiality of information assets.*

We've set up a firewall – UFW, on our server:

```
sudo apt-get install ufw
```

It's configured to allow traffic on ports 22 (SSH) and 80 (HTTP):

```
sudo ufw allow 22
sudo ufw allow 80
sudo ufw enable
```

Installing IPS:

***IPS** stands for Intrusion Prevention System. It is a security technology designed to monitor network and/or system activities for malicious or unwanted behavior and to take proactive measures to prevent potential security incidents. IPS operates by analyzing network and/or system traffic in real-time, identifying patterns or signatures associated with known threats, and responding promptly to block or mitigate the detected threats.*

We have implemented an Intrusion Prevention System (IPS) by installing Fail2Ban on our cloud server:

```
sudo apt-get update
sudo apt-get install fail2ban
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

It actively monitors log files for signs of malicious activity. In the event of suspicious behavior, Fail2Ban automatically takes measures to block offending IP addresses.

Setting strong Password Policy:

***A password policy** is a set of rules and requirements established by an organization to govern how users create and manage passwords for accessing computer systems, networks, or applications.*

We installed libpam-pwquality package:

```
sudo apt -y install libpam-pwquality
cracklib-runtime
```

and after the installation edited the `/etc/pam.d/`common-password file:

```
sudo vim /etc/pam.d/common-password
```

```
password requisite pam_pwquality.so retry=3
minlen=8 maxrepeat=3 ucredit=-1 lcredit=-1
dcredit=-1 ocredit=-1 difok=3 gecoscheck=1
reject_username enforce_for_root
```

options used:

- **retry=3:** Prompt a user 3 times before returning with error.

- **minlen=8 :** The password length cannot be less than this parameter
- **maxrepeat=3:** Allow a maximum of 3 repeated characters
- **ucredit=-1 :** Require at least one uppercase character
- **lcredit=-1 :** Must have at least one lowercase character.
- **dcredit=-1 :** must have at least one digit
- **difok=3 :** The number of characters in the new password that must not have been present in the old password.
- **gecoscheck=1:** Words in the GECOS field of the user's passwd entry are not contained in the new password.
- **reject_username:** Rejects the password if contains the name of the user in either straight or reversed form.
- **enforce_for_root:** Enforce pasword policy for root user

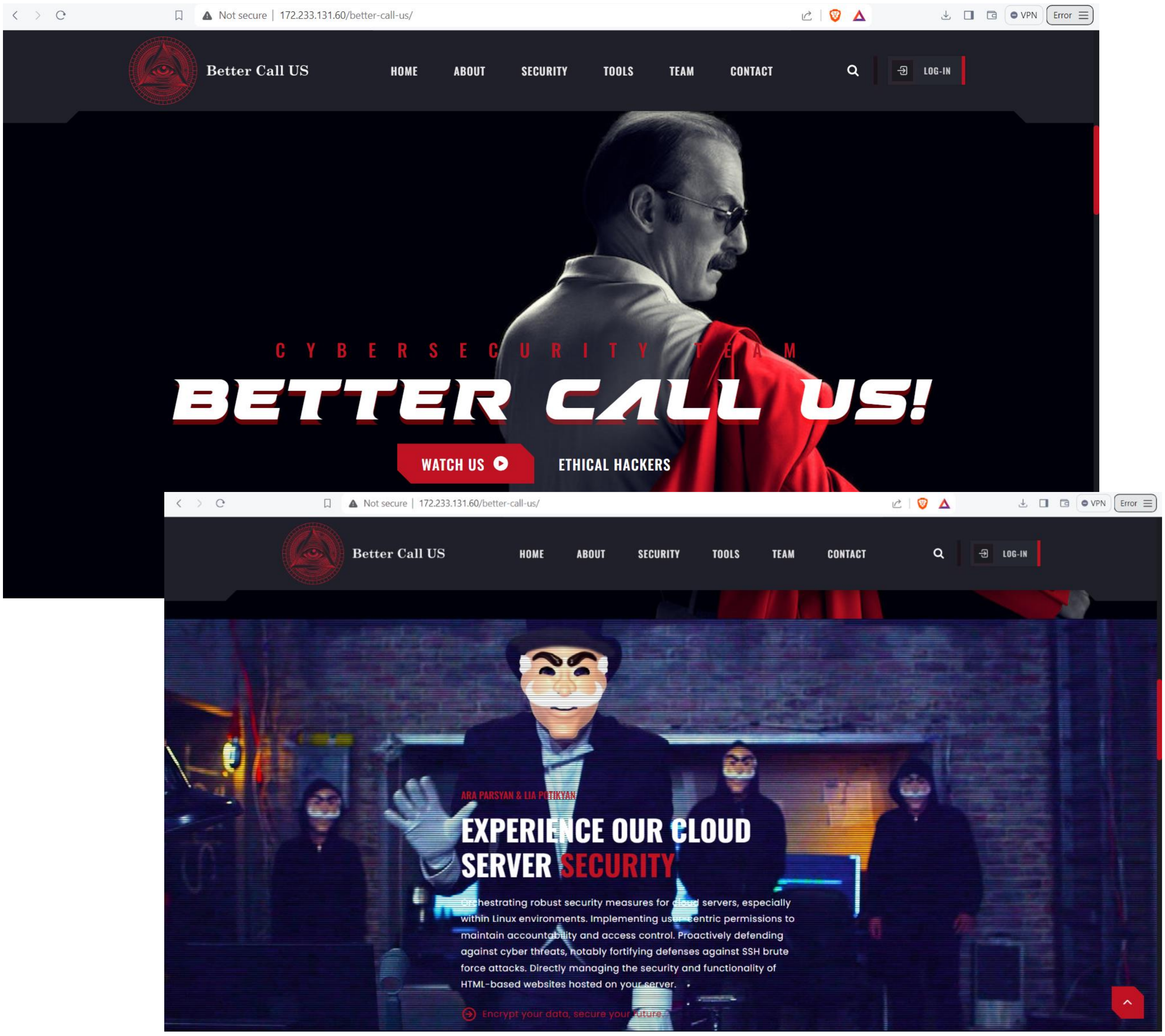
Installing Apache and hosting a website:

***Apache** is an open-source web server software widely used for serving websites. Known for its reliability and versatility, it supports various operating systems, features a modular architecture, and provides security measures like SSL/TLS support. Apache's configuration flexibility and ability to act as a reverse proxy or handle load balancing make it a popular choice for web hosting.*

We have installed apache on our ubuntu server:

```
sudo apt install apache2
sudo systemctl start apache2
sudo systemctl enable apache2
```

After that, we have successfully hosted our website:



ARA PARSYAN & LIA POTIKYAN

EXPERIENCE OUR CLOUD SERVER SECURITY

Orchestrating robust security measures for cloud servers, especially within Linux environments. Implementing user-centric permissions to maintain accountability and access control. Proactively defending against cyber threats, notably fortifying defenses against SSH brute force attacks. Directly managing the security and functionality of HTML-based websites hosted on your server.

Encrypt your data, secure your future

Setting up HTTPS with Nginx, obtaining SSL/TLS Certificate:

***HTTPS** is a secure version of HTTP, the protocol used for transmitting data between a user's web browser and a website. It adds a layer of encryption, ensuring that data exchanged between the user and the site is secure and protected from potential eavesdropping or tampering.*

***Nginx** is a high-performance web server and reverse proxy server that excels in handling concurrent connections. Widely used for hosting websites and applications, Nginx efficiently manages traffic, balances loads, and enhances the overall speed and reliability of web services.*

***An SSL** (Secure Sockets Layer) certificate is a digital certificate that establishes a secure and encrypted connection between a user's browser and a web server. It enables HTTPS by encrypting data transmitted over the internet, safeguarding it from unauthorized access.*

We installed certbot on our Ubuntu server:

```
sudo apt-get update
```

```
sudo apt-get install certbot python3-certbot-apache
```

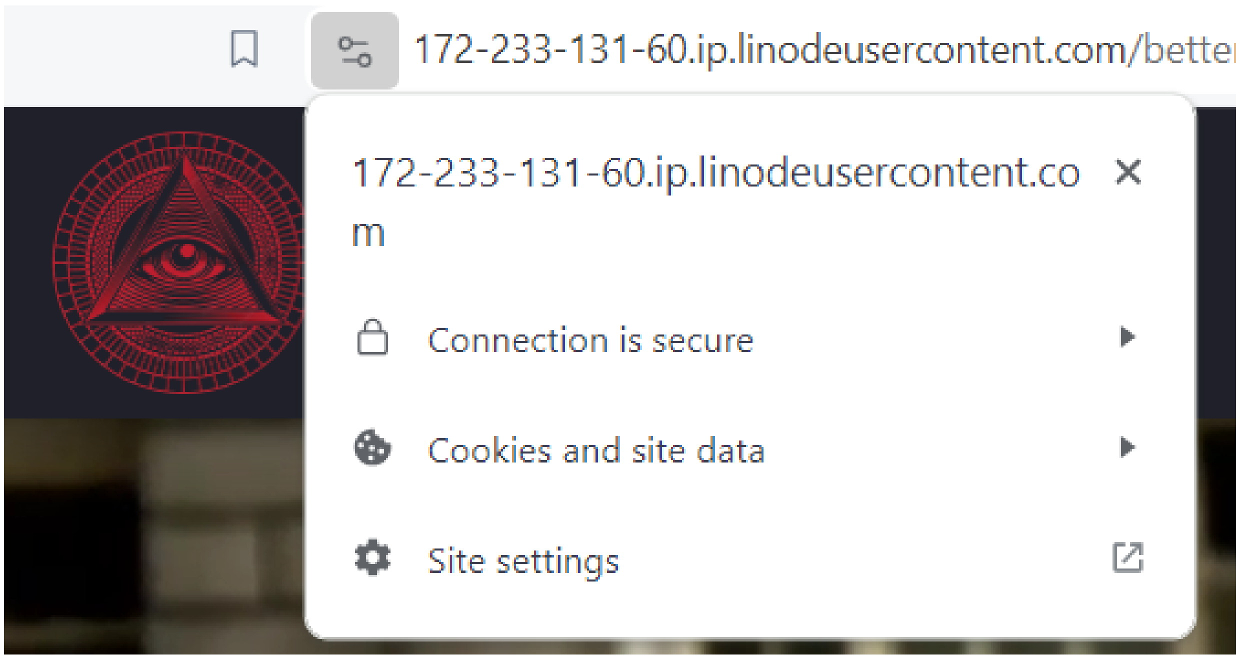
we obtained and installed an SSL/TLS certificate for our Apache web server:

```
sudo certbot –apache
```

and restarted:

```
sudo systemctl restart apache2
```

Now our website is accessible over a secure connection using the HTTPS protocol which provides an encrypted and secure communication channel between the user's web browser and the web server.


172-233-131-60.ip.linodeusercontent.com/bette

172-233-131-60.ip.linodeusercontent.com

Connection is secure

Cookies and site data

Site settings

Conclusion:

Through this project, we got hands-on experience in securing our cloud server and website hosting. Using keys, configuring the firewall, and implementing Fail2Ban taught us practical ways to enhance security. By configuring SSL/TLS certificates and Apache settings, we encrypted data transmissions, enhancing user privacy and trust. This not only makes our server safer but also gives us valuable insights into real-world measures for system protection, managing secure cloud infrastructure and website hosting.

References:

Akamai Cloud Manager https://www.akamai.com/

Fail2Ban Documentation https://www.fail2ban.org/

Github https://github.com/

Ubuntu https://ubuntu.com/

libpam-pwquality Documentation
https://linux.die.net/man/8/pam_pwquality

GeeksForGeeks https://www.geeksforgeeks.org/

ChatGPT https://chat.openai.com/

05/02/2024