Overlapping Trust Boundaries

Carmen Nicholson, Chris Conway, Doug Lindsay, Michael Towns, and Yasmine Hines  
College of Engineering and Science, Louisiana Tech University, Ruston, Louisiana 71272  
E-mail: {CAN024, CWC036, DCL020, MST039, YRH044}@latech.edu

***Abstract*— *In this project, we will discuss the vulnerabilities of overlapping trust boundaries and explore how to protect against them. Overlapping trust boundaries become an issue when a cloud provider shares their IT resources with multiple customers. We will experiment by casting each member as cloud consumers and one member as the cloud provider. We will then ask the members to attempt to retrieve data belonging to another member. The cloud provider will also be tasked with retrieving data from the consumers as that is another possible vulnerability. The goal of this project is to design a system in which the cloud consumers cannot retrieve each other's data and the cloud provider cannot view the data.***

# Introduction

Overlapping of trust boundaries involves one cloud provider and multiple different cloud consumers or users. With cloud services sometimes sharing resources among many different clients, they must take care of the boundaries that have been established between them. It provides attackers with opportunities to threaten IT resources that are shared between multiple cloud consumers. An attack commonly used is guest-hopping, this is where an attacker possibly has access to an Operating System (OS) and uses its access to compromise a different OS within the same cloud system. An ideal method for diminishing this threat is to provide traffic isolation.

Our solution would be to not allow other members of the group to be able to retrieve data from other members by adding multiple layers of security. With adding security, this allows the user who owns the directory will be the only one to be able to read from and write to the files in their directory. Each user will have the opportunity to do this using the chmod command.

We are using Amazon’s EC2 in order to host live Ubuntu instances configured with users for each member of the group.

# Related Work

Cloud computing gives many opportunities for large scale computing projects at lower costs than if implemented by a single user [3]. However, as more resources are distributed it becomes difficult to track where different resources are used creating an issue of compliance with local regulation.

Data privacy is another concern because the data is often stored on machines owned by different organizations and if it is not secured properly can lead to breaches of privacy.

Security is problematic because it relies on all participants to be fully protected. If one organization is breached, the security of all others working together may be compromised. With more ways to access the data, it becomes increasingly likely that an attacker can find a vulnerability.

Pearson, Siani, and Azzedine Benameur offer a few suggestions as to how these challenges might be handled after a breach [3]. It is suggested to carefully define what organizations are in charge of what parts of the security, accountability in the case of breaches or data loss. However, these measures are reactive and do not actually answer how security will be addressed.

Clients do not have direct control over the cloud which leads to data privacy issues [4]. The users rely on cloud providers to protect their information from being stolen or the cloud providers from leaking their information. The cloud providers us cryptography to protect their user's data. Cryptography alone cannot fix this issue. The users of the cloud services will need to rely on other forms of privacy enforcement to protect their data. Privacy preservation may be a key or a way to help protect the cloud provider’s clients’ information.

# Background

In order to design our system, we used an Amazon Elastic Compute Cloud (EC2). EC2 is a web service that provides secure and scalable computing capacity within Amazon Web Services (AWS). Amazon makes it simple to create an account and use multiple cloud resources for free. We were able to use this to host our operating system. For our OS, we opted for an Ubuntu instance. Ubuntu is a complete Linux operating system that is freely available to use.

Once the OS was up, each user was set up using Linux commands through the command-line. Instructions on how to do this were available on Amazon. In order to gain access to the server, each user had to use secure shell (SSH) and use their specific private key. SSH is a network protocol that gives users the ability to have a secure way to access a server over an unsecure network. For this to happen, all that was needed was knowledge of commands used through the terminal or command-line interface.

After the user had logged into their instance, It was time to focus on the security aspect. Each user first needed to change their permissions in order to make their directory private to them using terminal or command-line interface. Using SSH, the user also had the capability to add another layer of security by creating jail. This means that the user could isolate their process and children form the rest of the system. Because none of the user’s processes were running as root, it made them much harder to break out of. Each user would now using their home directory as the root meaning only the user was able to read/write in their own home directory.

# Methods

Starting with security, we’re using Amazon’s EC2 in order to host live Ubuntu instances that can be accessed by anyone on the team. Every member will receive their own user accounts, along with private keys assigned to each member. The anticipated result is that, outside of using hacking tools, each user folder will only be accessible from the member assigned to it. The added layer of using private keys alongside usernames and passwords ensures that, in order to access a user’s folder, not only will they need the user info, they will need the necessary private key. The first step is to get an account set up for every member of the team that they can ssh into.

## *User accounts*

Amazon has directions on how to add users to Linux instances [1]. First, is to add a new user account to the instance using “sudo adduser new\_user”. Second, we need to switch to the new user so that the files created later on have the correct permissions using “sudo su new\_user”. Next, we need to create a .ssh directory that only new\_user has read/write/open access to. This is accomplished using three commands in sequence, “cd” to go to new\_user’s home directory, “mkdir .ssh” to create the .ssh directory, and “chmod 700 .ssh” to set the permissions to the directory. Lastly, we need to create an authorized\_keys file in the .ssh directory. We use a string of commands to accomplish this, “cd” to go to new\_user’s home directory, “touch .ssh/authorized\_keys” to create the file, and “chmod 600 .ssh/authorized\_keys” to set the permissions on the file. This process will need to be repeated for every member of the team.

The next step is to allow everyone to ssh into the instance. To accomplish this we created a key pair for every member of the team and handed out the private key to the member it belongs to. Once the key pairs are made and downloaded we get the public key using the “ssh-keygen -y” command and give the private key file “New\_user.pem” when prompted. Once we had the public key we copy it from the terminal and paste it into the authorized\_keys file on the instance using vim. Once the file is saved, every member of the team will be able to ssh into the instance.

## *Adding Security*

Now that everyone can get into the instance, security will need to be added that prevents users from modifying and seeing each other's data.

In order to prevent the user from modifying each other's data, each users home directory will have their permissions changed to 700 using the chmod command, meaning only the user who owns the directory will be able to read from/write to files in that directory. This will force the user to only be able to read/write to their own home directory.

In order to prevent users from seeing each other data, one method is to create a chroot jail. This makes the user’s home directory act like root which means they will not be able to navigate out of it. Allan Feid has an article explaining how to make ssh jails and how they are useful [2]. The first step is to create a group for the users, “sshusers”, using the addgroup command, “addgroup -g sshusers”. After adding every user to the group using “usermod -a -G sshusers new\_user”, we need to emulate the root directory by copying the necessary folder from root to the users new root folder along with other necessary files. This can be accomplished using this string of commands:

mkdir -p /home/{user}/{dev,etc,lib,lib64,usr,bin}

mkdir -p /home/{user}/usr/bin

chown root.root /home/{user}

mknod -m 666 /home/{user}/dev/null c 1 3

cd /home/{user}/etc

cp /etc/ld.so.cache ./

cp /etc/ld.so.conf ./  
 cp /etc/nsswitch.conf ./

cp /etc/hosts ./

Lastly, we need to configure sshd to chroot the users of the group sshusers. This can be done by adding the following script to the end of the sshd\_config file in etc/ssh/:

Match group sshusers  
 ChrootDirectory /home/%u  
 X11Forwarding no  
 AllowTcpForwarding no

After completing these steps, the Ubuntu Linux instance will be setup for experiments.

# Experiments

After setting up the environment for these experiments, the team will spend some time getting access to

# Results

[Replace this with Your Results section.]

# Discussion

[Replace this with Your optional Discussion section.]

# Conclusions

[Replace this with Your Conclusions section. Remember to include your team members’ ideas for future work.]

##### Appendix A

[Replace this with Your Source Code and Description.]

##### Appendix B

[Replace this with Your Team Member Contributions breakdown.]

##### References

1. “Add New User Accounts with SSH Access to a Linux Instance.” Amazon, Amazon, 2017, aws.amazon.com/premiumsupport/knowledge-center/new-user-accounts-linux-instance/.
2. Feid, Allan. “ Creating a Chroot Jail for SSH Access.” *Allan Feid*, 2018, allanfeid.com/content/creating-chroot-jail-ssh-access.
3. Pearson, Siani, and Azzedine Benameur. "Privacy, security and trust issues arising from cloud computing." *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010.
4. Van Dijk, Marten, and Ari Juels. "On the impossibility of cryptography alone for privacy-preserving cloud computing." *HotSec* 10 (2010): 1-8.