**Ethical Hacking & Cyber Security - हिंदी Essentials**

**DAY 8 ASSIGNMENT**

Question

1. Create a user named bob in Kali.
2. Enable ssh on Kali
3. Allow bob to login into kali from git bash without a password using keys.

**ANS 1.**

Create a user named bob in Kali.

Kali Linux is a fantastic operating system for penetration testing and security evaluation. It comes with virtually all security tools built in, it’s lightweight by default, and it has a huge ecosystem that is constantly helping with the project.

*For instructor-led Security training see our*[course schedule](http://www.interfacett.com/course-schedule/#cissp-ceh-pki-security)*.*

I created a new installation of Kali Linux recently. One of the first post-installation tasks I did was to create a new user for daily use. Sadly, Kali only creates a default root user during setup. Running as root all the time is a horrible security practice, so I recommend that you create a new user as soon as possible after installation.

To create a new user in Kali Linux, first pop open a Terminal window.

Knowing how to use Linux to add a user to a group is fairly simple once you know the commands. Whether you need to do any of the following, we’ve got your answer:

Add a user to Linux

Delete a user in Linux

Add a new group to Linux

Add a user to a group in Linux

Change a password

Modify a user in a Linux group

But first, let’s cover the key files that are associated with this process in case you are new to Linux or are working on passing a certification exam. Then, we’ll give you a step-by-step guide for each scenario you may encounter.

It’s important to introduce two files that we will be using as examples of adding users to a group in Linux.

In the /etc directory, the passwd and the group files

**ANS 2.**

Enable ssh on Kali

The Linux distribution Kali used by many penetration testers (including those here at [LMG Security](https://lmgsecurity.com/)) recently released version 2017.1 of their rolling release. For quite some time now (Since version 2.0) Kali has used Systemd (System Management Daemon) in place of an init system.  This change brought with it a new way of enabling and starting services, even though many still use the old commands, which often still work but may also lead to errors.  This post will go over the Systemd method for enabling and starting the SSH (Secure Shell) service on Kali Linux.

# apt list openssh-server

You should see the version with [installed] after it like this:

https://www.lmgsecurity.com/wp-content/uploads/2019/12/SSHKali-300x11.jpg

If it’s not installed, you can use this command to install it:

# apt install openssh-server

When enabling the service, be sure to fully secure SSH first.  I will cover some of the basics briefly, but this is not meant to be a guide on securely running an SSH server.  Since Kali comes with pre-generated SSH keys, to make it more secure, the first thing we will do is generate new ones.

To backup the original keys first as a precaution use:

# mkdir /etc/ssh/default\_keys

# mv /etc/ssh/ssh\_host\_\* /etc/ssh/default\_keys/

Then to regenerate the keys:

# dpkg-reconfigure openssh-server

It’s useful to know that Systemd has different units, a unit configuration file encodes information.  The units relevant to SSH are ssh.service and ssh.socket. At a basic level a service unit controls a process and a socket unit controls a filesystem or network socket

# systemctl stop ssh.socket

**ANS 3.**

Allow bob to login into kali from git bash without a password using keys.

If you use the SSH transport for connecting to remotes, it’s possible for you to have a key without a passphrase, which allows you to securely transfer data without typing in your username and password. However, this isn’t possible with the HTTP protocols – every connection needs a username and password. This gets even harder for systems with two-factor authentication, where the token you use for a password is randomly generated and unpronounceable.

Fortunately, Git has a credentials system that can help with this. Git has a few options provided in the box:

* The default is not to cache at all. Every connection will prompt you for your username and password.
* The “cache” mode keeps credentials in memory for a certain period of time. None of the passwords are ever stored on disk, and they are purged from the cache after 15 minutes.
* The “store” mode saves the credentials to a plain-text file on disk, and they never expire. This means that until you change your password for the Git host, you won’t ever have to type in your credentials again. The downside of this approach is that your passwords are stored in cleartext in a plain file in your home directory.
* If you’re using a Mac, Git comes with an “osxkeychain” mode, which caches credentials in the secure keychain that’s attached to your system account. This method stores the credentials on disk, and they never expire, but they’re encrypted with the same system that stores HTTPS certificates and Safari auto-fills.
* If you’re using Windows, you can install a helper called “winstore.” This is similar to the “osxkeychain” helper described above, but uses the Windows Credential Store to control sensitive information. It can be found at [https://gitcredentialstore.codeplex.com](https://gitcredentialstore.codeplex.com/).

You can choose one of these methods by setting a Git configuration value:

$ git config --global credential.helper cache

Some of these helpers have options. The “store” helper can take a --file <path> argument, which customizes where the plaintext file is saved (the default is ~/.git-credentials). The “cache” helper accepts the --timeout <seconds> option, which changes the amount of time its daemon is kept running (the default is “900”, or 15 minutes). Here’s an example of how you’d configure the “store” helper with a custom file name:

$ git config --global credential.helper store --file ~/.my-credentials

Git even allows you to configure several helpers. When looking for credentials for a particular host, Git will query them in order, and stop after the first answer is provided. When saving credentials, Git will send the username and password to all of the helpers in the list, and they can choose what to do with them. Here’s what a .gitconfig would look like if you had a credentials file on a thumb drive, but wanted to use the in-memory cache to save some typing if the drive isn’t plugged in:

