**NAME *: LAKSHITHA RAJ VASANADU***

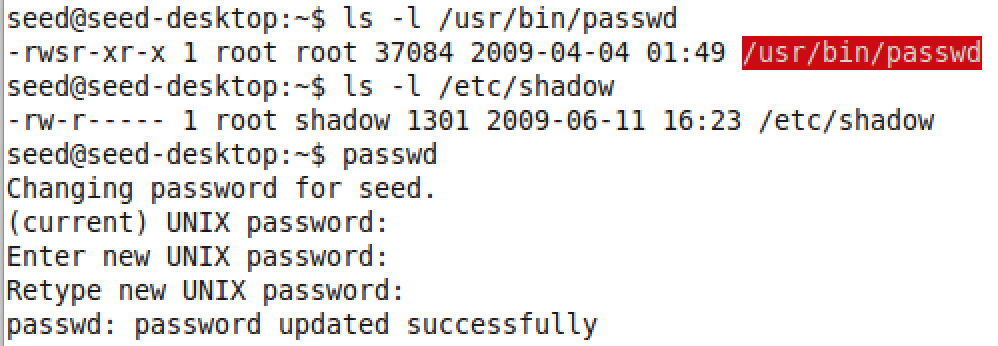
**ID *: 00001115006***

**LAB ASSIGNMENT 1**

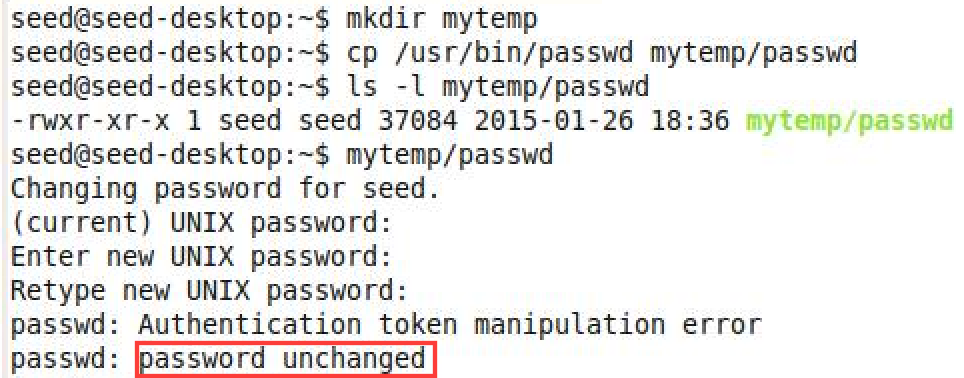
**Set-UID Program Vulnerability Lab**

1. “**passwd**”, “**chsh**”, “**su**” and “**sudo**” commands need to be Set-UID programs.

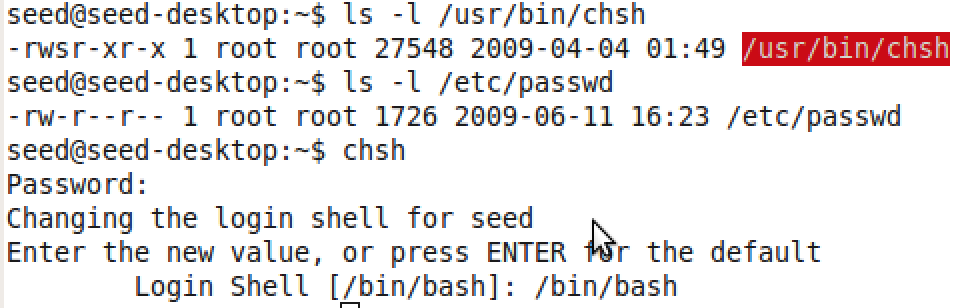
* **passwd**
* This command allows the users to change their passwords. It is runnable by any user.
* The users’ passwords are stored in “**/etc/shadow”** file which is neither readable nor writable to normal users. It is a root-owned file.
* It gives the user temporary privilege to change the password so that the user will not get access to other root-owned files/programs and hence should be a Set-UID program.
* When the “passwd” program runs, the effective user id for the user will be root.

****

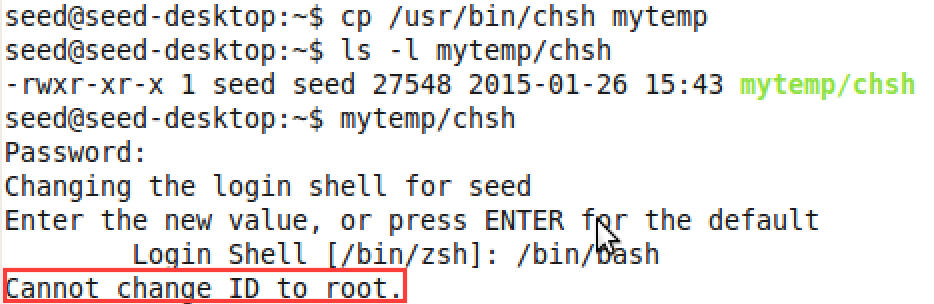
* If the program is not Set-UID, the users can’t change their password, as the program will not be able to access the root-owned files. If the files are not root-owned, all users will be able to access them that threaten the security.

****

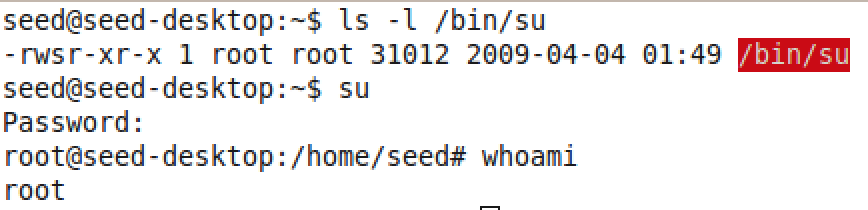
* **chsh**
* This command allows the users to change their login shells.
* The login shell details of the users are stored in **“/etc/passwd”** which is a root-owned file.
* It is a Set-UID program as it ensures that users can modify their own login shells temporarily by getting root access.



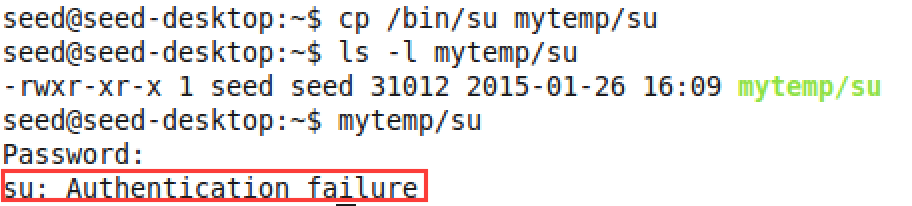
* If the program is not Set-UID, the normal users will not be able to change their own shell. Or, they might be able to access all the files owned by root.

****

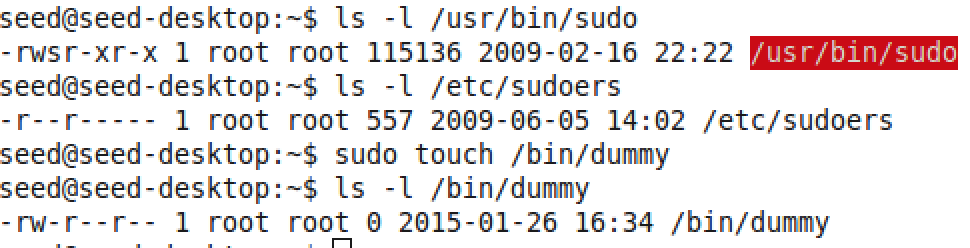
* **su**
* This command allows the user to become the super user to perform operations that have root privilege.
* User must pass super-user’s password as an argument.

****

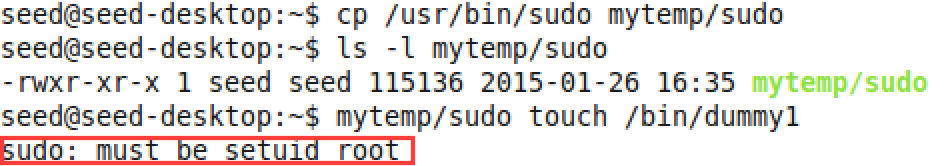
* If the program were not Set-UID, then a non-super user would not have privilege to switch and would be running as original one only or would have been able to exploit all the other users. It needs to access files like “/etc/shadow” which are root-owned.

****

* **sudo**
* This command allows the user to become super user or another user as specified in **“/etc/sudoers**” file which is root-owned.
* The local and effective uid/gid are set to match those of target user as specified in the passwd file.

****

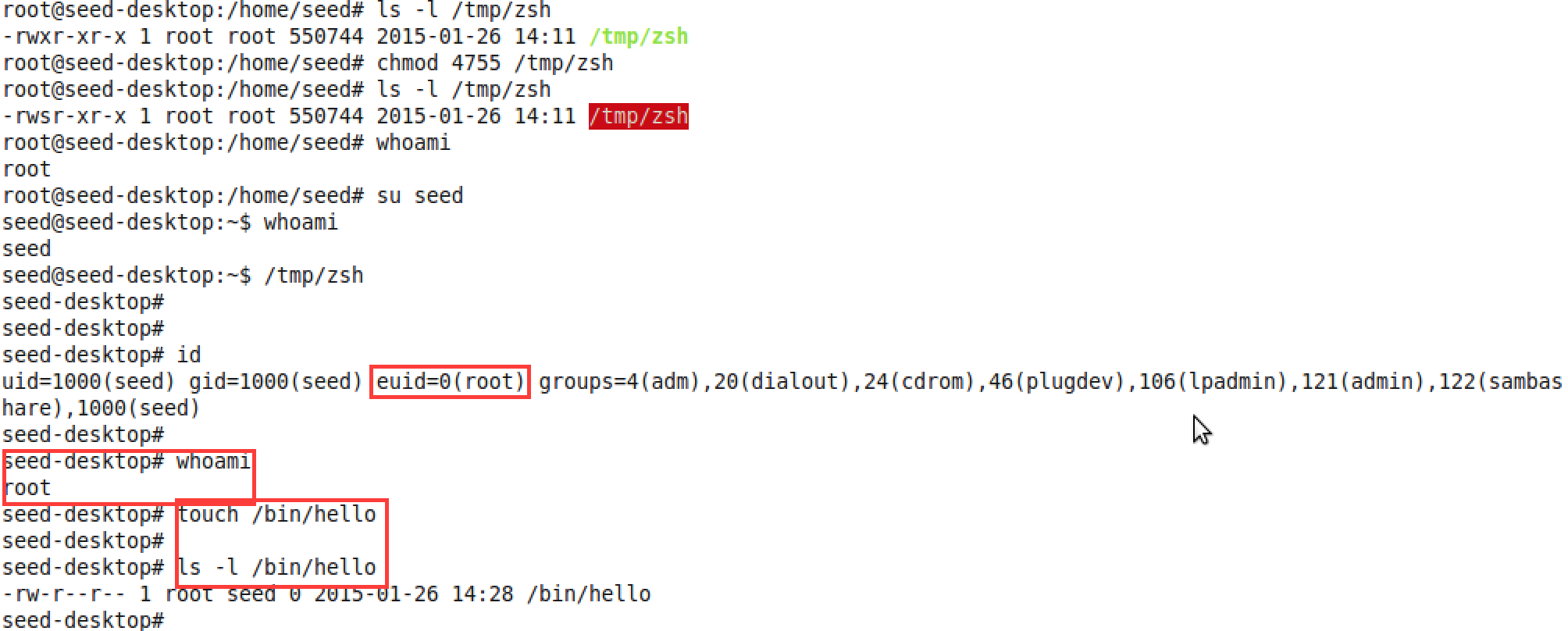
* If it was not a Set-UID program, then ordinary users will not be able to gain privileges of other users nor that of the root.

****

2) Bash and Zsh

1. **Zsh**

* Login as root : ***su***
* Copy zsh to tmp : ***cp /bin/zsh /tmp***
* Set /tmp/zsh as Set UID program: ***chmod 4755 /tmp/zsh***
* Switch to normal user: ***su seed***
* Check the user : ***whoami*** => seed
* Run **/tmp/zsh**



* The normal user **gets the root** privilege. As illustrated above, user ***seed*** creates a file ***hello*** in root-owned **/*bin*** directory.
* The effective user id is that of the root.

1. **Bash**

* Login as root : ***su***
* Copy bash to tmp : ***cp /bin/bash /tmp***
* Set /tmp/bash as Set UID program: ***chmod 4755 /tmp/bash***
* Switch to normal user: ***su seed***
* Check the user : ***whoami*** => seed
* Run **/tmp/bash**

****

* The normal user **does not** **get the root** privilege. As illustrated above, user ***seed*** fails to create a file ***hello*** in root-owned **/*bin*** directory.
* The effective user id is same as the real user id.
* ***Bash has better security than Zsh. Bash immediately drops the privilege as soon as it detects that the program is running with setuid on.***

4) system() and PATH

Let the given program be **sys.c**. It is compiled into **sys**.

*#include <stdio.h>*

*int main() {*

*system(“whoami”);*

*system(“ls”);*

*return 0;*

*}*

Let the program for exploitation be **my\_ls.c**. It is compiled to **ls**.

*#include <stdio.h>*

*int main() {*

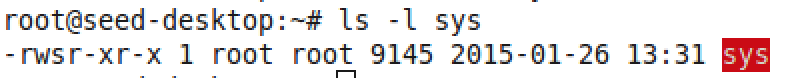
*system(“whoami”);*

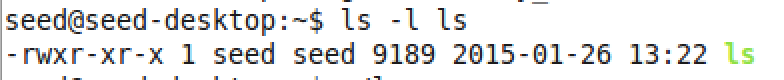
*system(“touch /bin/exploit”); //Creates a file in root-owned directory.*

*return 0;*

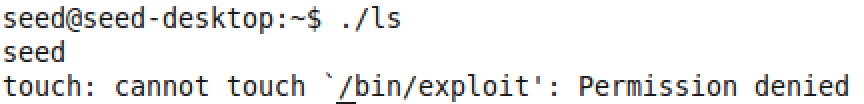
*}*

The permissions of the files are as follows:



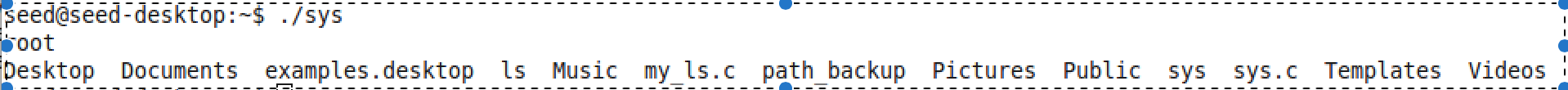


When a normal user like *seed* runs the “*my\_ls.c*” directly, the user does not have permission to perform the embedded operation.

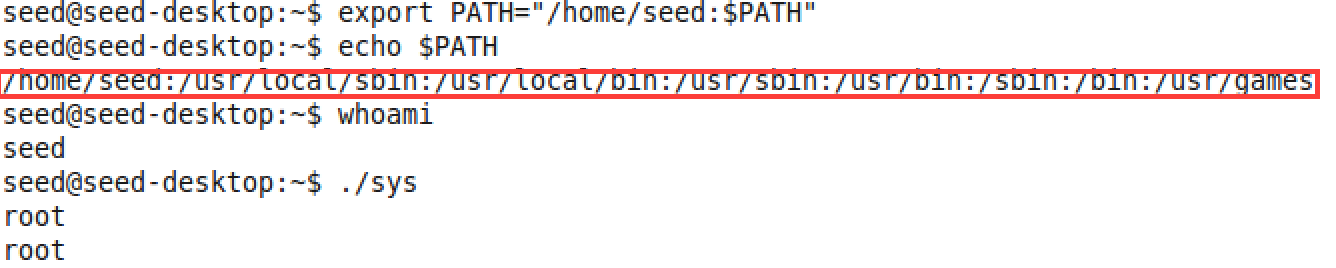


1. **Zsh**

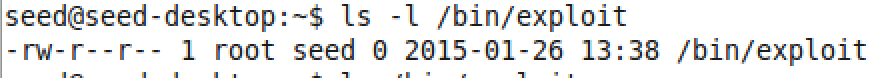
* When a normal user like seed runs “sys.c” directly, the user gains the root privilege as it is a set UID program. The program outputs the output of “/bin/ls” command.



* This **sys.c** program can be exploited to call “**my\_ls.c**” by changing the value of **PATH** env variable as follows:

****

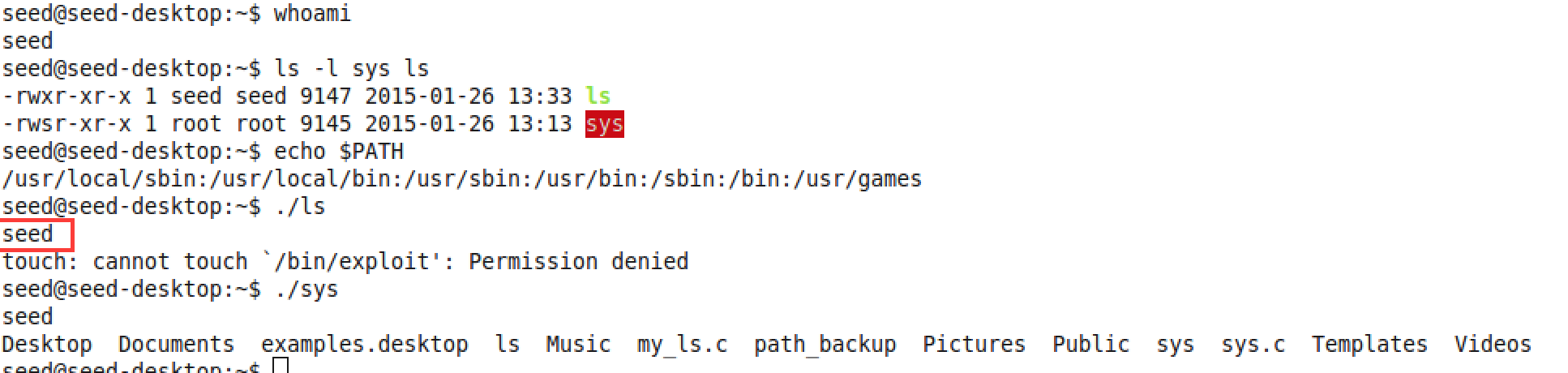
* As illustrated above, the user-defined program is run which creates a file in the root-owned **/bin** directory. It gains root as effective user id.

****

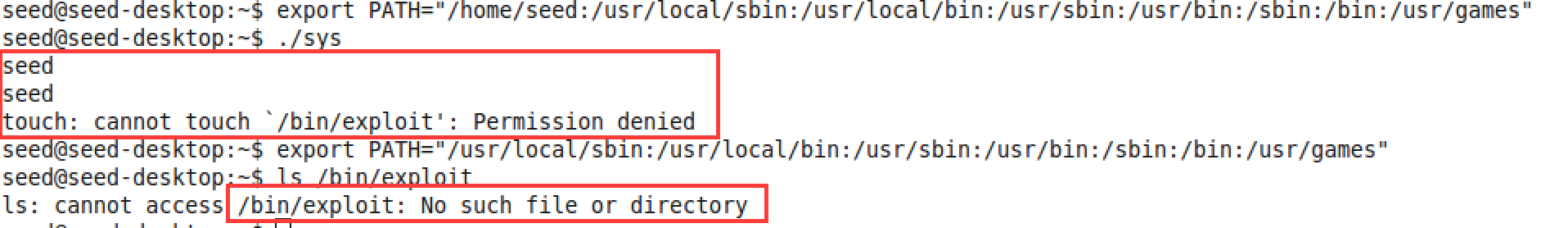
* **Thus, the set UID program can be made to run user-defined code. Zsh exploits setUID mechanism and leads to a vulnerability. The user code gains root access and this can be dangerous as it can run malicious code or delete the important files.**

1. **Bash**

* When a normal user like seed runs “sys.c” directly, the user does not gain root privilege even though it’s a setUID program. The programs outputs the output of “/bin/ls” command. Bash drops the privilege once it detects that the program is a setUID program.



* We can try to exploit the above code by modifying PATH variable :



* But we find that exploit can not be made as the program does not run with root privilege.
* **Thus, bash does not exploit SetUID program and hence provides more protection. The code does not have root access.It drops the privilege once it detects that the program is a setUID program.**