Vulnerability Research

A core task on the Ubuntu security team is to research a given vulnerability, determine whether it affects software distributed by Ubuntu or Canonical, and if so, remediate the issue. Like other Linux distributions, we usually do this by backporting a cherry-picked fix to the existing version we distribute, rather than updating to a whole new version.

For this task, please research CVE-2016-0741:

- Identify what software this issue affects,
- What is the commit or commits to fix it, if any are available
- Describe how would you go about determining whether any Ubuntu releases are affected and which ones,
- Describe how you would address the vulnerability,
- Describe how you would verify that your fix does address the issue and does not introduce regressions.

Design Review

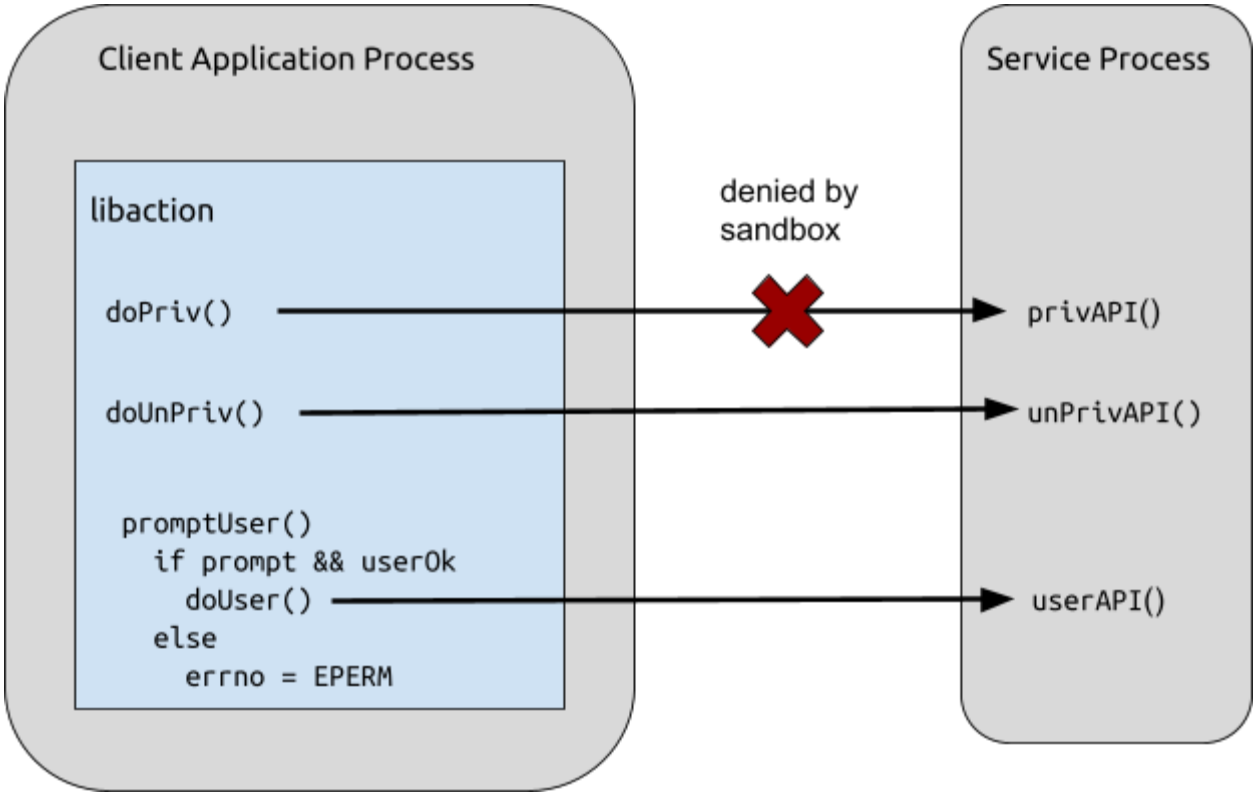
As part of the security team, sometimes you may be asked to participate in design reviews. Consider if someone asked you to review a design where an untrusted client application that is running in an existing sandbox environment is meant to safely interact with a trusted service application over some local IPC mechanism. Assume that the sandbox is capable of mediating API calls between the client and service, separate from either.

Given the following function definitions in the service process:

- `privAPI()` - performs privileged actions on behalf of an application
- `unPrivApi()` - performs unprivileged actions on behalf of an application
- `userAPI()` - performs actions on behalf of the application if the user approves when prompted

What is the design flaw in the following diagram?

How would you suggest fixing it?



Code Audit

Another task the security team will get asked is to perform code reviews. With the following code, answer the following questions:

- What errors are in this code?
- What threat model is appropriate for this code?
- What threat model would you consider if this program were exposed to the internet?
- What test cases would you use to test this program?
- Bonus: rewrite this code to address the issues you've identified. What choices need to be addressed with the rewrite? What choices would you make to cope with future requirements changes? What changes in your choices if this program is a simple tool for yourself versus if this is part of a larger product you're distributing to others?

```
$ cat greetings.c
/*
 * Compile this program with 'make greetings'; run it with ./greetings
 *
 * A simple program to say hello and get to know the user better.
 *
 */

#define _GNU_SOURCE
#pragma GCC diagnostic ignored "-Wformat"
#pragma GCC diagnostic ignored "-Wformat-security"
#pragma GCC diagnostic ignored "-Wimplicit-function-declaration"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char upper(char c) {
	return c & ~0x20;
}

char lower(char c) {
	return c | 0x20;
}

char* getinput(char *prompt) {
	char input[32];
	char *ret;

again:
	printf(prompt);
	if(!gets(input))
		goto again;

	ret = malloc(strlen(input));
	strcpy(ret, input);
	return ret;
}

char *uppercase(char *s) {
	int i;
	char *o = malloc(strlen((s)));
	strcpy(o, s);

	for (i=0; i<strlen(o); i++) {
		o[i] = upper(o[i]);
	}

	return o;
}

int main(int argc, char* argv[]) {
	char *name;
	char *friendlyprompt;
	char *icecream;
```

```
        char *pizza;
        char *filename;
        char *command;

        name = getinput("Welcome, what is your name: ");

        asprintf(&friendlyprompt, "%s, what is your favourite ice cream: ",
                        name);

        icecream = getinput(friendlyprompt);

        asprintf(&friendlyprompt, "%s! I love %s ice cream too. What's "
                        "your favourite pizza, %s? ",
                        uppercase(icecream),
                        icecream,
                        name);

        pizza = getinput(friendlyprompt);

        asprintf(&friendlyprompt, "%s is a good pizza topping.\n", pizza);
        printf(friendlyprompt);

        asprintf(&filename, "/tmp/%s", name);
        asprintf(&command, "echo 'It was a pleasure to meet you, %s' >> %s",
                name,
                filename);

        asprintf(&friendlyprompt, "I've left you a present in %s\n", filename);
        printf(friendlyprompt);

        system(command);

        exit(0);
}

$ make greetings
cc     greetings.c   -o greetings
/usr/bin/ld: /tmp/ccfkFCiL.o: in function `getinput':
greetings.c:(.text+0x69): warning: the `gets' function is dangerous and should not be used.
$ ./greetings
Welcome, what is your name: sarnold
sarnold, what is your favourite ice cream: vanilla
VANILLA! I love vanilla ice cream too. What's your favourite pizza, sarnold? pepperoni
pepperoni is a good pizza topping.
I've left you a present in /tmp/sarnold
$ cat /tmp/sarnold
It was a pleasure to meet you, sarnold
$
```