Security Engineering - Winter 2017

Sambuddho Chakravarty

January 25, 2017

Assignment 1 (Total points: 40)

Due date: February 12, 2017. Time: 23:59 Hrs.

ACLs and Setuid

The objective of this assignment is to familiarize you with using setuid() (and related) system call(s). In this assignment you would need to use your existing laptop and desktop and create multiple REAL users -– lets call them "larry", "bill", "steve", "mukesh", 'azim, 'mark' etc. (whatever you feel like). You would add these using the actual `adduser` command. Their home directories should be inside the simple_slash directory that you created earlier. The directories could have names like simple_slash/home/larry (for user larry). The /etc/passwd file should reflect these users and their home directories. You also need a secondary 'root' users, lets call it 'fakeroot'. The directories 'simple_slash' and 'simple_slash/home' should have 'fakeroot' as their owner.

You need to implement ACLs, through `setuid`, which should override DAC. These ACLs would be stored with the files (as a header). One way to do this is to use a C(C++) structure/class that has an array of strings representing ACLs, along with a pointer to a character buffer which can be modified based on the number of ACL entries in the file. *E.g.*

```
struct file_data{
unsigned int acl_len; // Number of acl strings
unsigned char **acl; // ACL strings
unsigned int data_len; // Number of bytes of 'data'
unsigned char *data; // File data
}
```

You may borrow concepts from what you understand of Linux ACLs. Access to the ACL would be via programs called 'setacl' and 'getacl'. These would be individual binaries with their `setuid` bit turned on and their default owners would be 'fakeroot'. Whenever these programs are invoked they determine the actual user ID of the invoking program using `getuid()` system call. Thereafter the program needs to check the ACL entries of the file to determine if at all the user is allowed to modified them or not. If the program decides to grant access to the invoking program to modify the ACLs, it must switches the user ID to the actual owner of the target file (using `seteuid()` family of system calls) and proceeds with the ACL modifications. Else, the program fails gracefully, printing a permission denial message.

You also need to implement individual programs for 'fput', 'fget', 'cd' and 'create_dir' which would also be setuid programs, owned by 'fakeroot'. Similar to the 'setacl' and 'getacl' programs, these programs also check the invoking user ID and determine if it has permissions to perform the operation corresponding to the program or not. The default ACL values for a newly created file should be same as the DAC bits. However, the subsequent access to the file must be via the above programs and must be mediated through the ACLs.

You need to check for every corner case with regards to the functionality. Feel free to consider other possible assumptions. DO NOT forget to list the assumptions in the system description that you would submit.
Note: This assignment assumes real users (who have real UIDs), with real directory structures. You are supposed to write real programs, corresponding the aforementioned commands, just like you have in a real *nix system.

Grading Rubric

- Successful compilation using Makefile – 5 points.
- Use of system calls like – setuid() to implement the aforementioned functionality, along with the system calls used previously like readdir(), opendir(), read(), write() etc. – 10 points.
- Successfully handling various access control scenarios – e.g. users attempting to access a file which the DAC doesn't allow by the ACLs allow or vice versa – 10 points.
- Successfully defending against atleast 3 attacks/bugs/errors – 10 points (List the bugs/errors/attacks that you defend against).
- Description of the systems, commands to execute and test the program and the assumptions that you made – 5 points.

Late Submission Policy

- Submitted on or before February 12, 2017 (23:59 hrs) – No points deducted.
- Submitted after February 12, 2017 but on or before February 14, 2017 (23:59 hrs) – 5 points deducted.
- Submitted after February 14, 2017 – 10 points deducted.