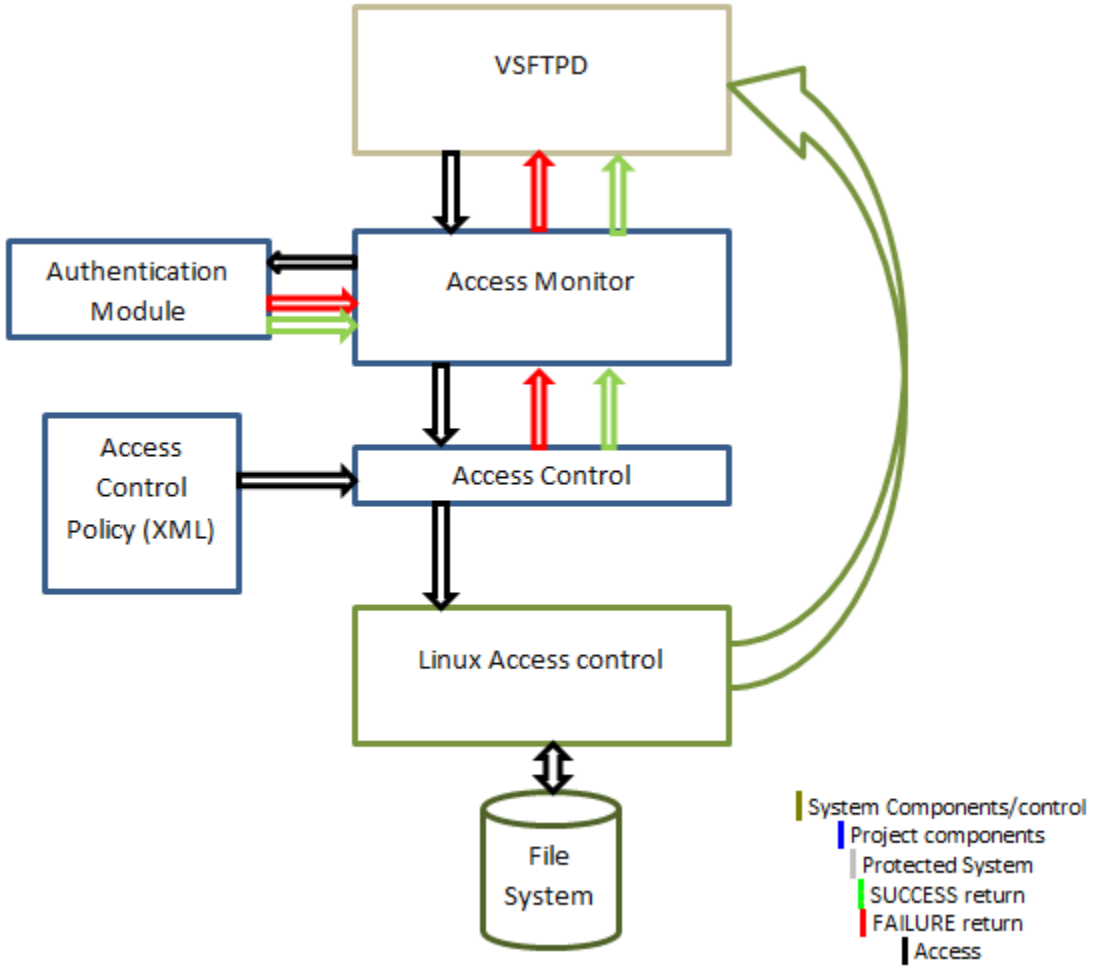
GT File Access Monitor

I. DESIGN



The user connects to the server either remotely or locally using VSFTP. Each time a new connection is made first the user is authenticated by using authentication module if the authentication is unsuccessful, the process will be terminated. When the traced process does a system call, call specifications and user details are passed to the Access Control and then the Access Monitor will either grant or deny the system call depending on the response received from the access monitor.

II. IMPLEMENTATION

The system has been divided in three modules which have been integrated together. The core modules are Access Monitor, Access Control, and Authentication. The Access Monitor and Access Control are tied together in implementation, whereas the Authentication module is a separate logical entity. The implementation details of each module are described below:

- **Access Monitor:** The access monitor will check all remote accesses made to the file system by the VSFTP program. This is done using PTRACE to trace system calls made by the vsftpd process. The monitor will intercept all the relevant system calls and will determine the result of the action by using the Access Control and Authentication modules. If either of these modules denies access to the user, the monitor will terminate the call by modifying the registers so that the user is denied access to the file or folder. **The access monitor is self-replicating; if the process being traced clones or forks to create a new process access monitor will clone itself to attach to the newly forked process.** The access monitor is designed agnostic to the process being traced (here it is vsftpd), it can check all the system calls being traced and also any ways in which files can be accessed i.e by creating symbolic links , hard links and renames and other side channels.

- **Access Control:** The access control module is used to implement an Access Control Policy which allows the administrator to specify which user is Allowed/Denied access to what files/folders and in what mode. For example, the administrator can disallow user X write permission on file Y or disallow group G read permission on file Z. **The entire policy will be written in an XML file which is parsed by the module into an ACL. Depending on the access made by the current user, the module will either grant or deny permission to the user. Negative/Deny rules take precedence over Positive/Allow rules.** By default, if the user/file is not present in the policy file, the module will grant access to the user. The

access control module is called before the access control check performed by the system.

- **Authentication:** The authentication module implements the technique mentioned in the Password Hardening paper. The feature vector is read from a file and if the correct hardened password can be calculated, the user will be granted permission. When the user registers the first time, the module will initialize the user's parameters and will create a new random hardened password for the user. The next time the user logs in; the program will read values from the previously created Instruction table and will use this to decrypt the history file. If the history file is decrypted successfully, the module will update the Instruction Table and History file and encrypt them with the weak and hard password respectively. The new values are calculated based on the history of the user's login pattern. The last 20 feature vectors are used to calculate the average for updating the instruction table. The module will start updating the instruction table with garbage values only after 18 entries. Before that it will allow the user login with either alpha or beta selected. **This module is called at the start of executing process handling the user request. If authentication fails the process will be terminated.**

III. INTEGRATION

The modules are integrated in the following manner:

The primary module is the Access Monitor which performs the task of tracing system calls and accordingly calls the Access Control module which will either grant or deny permission to the user. The Authentication module is called by the monitor every time we find out that the user is the owner for a file. The Access Monitor and Access Control modules have been created as a single logical module. The Authentication module is a separate logical entity which provides access to the interface via a static library.

IV. PERFORMANCE

There is a negligible performance impact on the VSFTPD program. Although a slight delay is encountered during login of the user, the user will not experience any kind of delay during file access.

V. EVALUATION

We tested our code in many scenarios and also used the Flaw Finder tool for Vulnerability Assessment. We present our findings below:

The findings of vulnerability assessment tools are traced to be false positives because all the data used for unsafe strings functions like: strcpy, strlen etc. are all operating on untainted data which was sanitized before calling the functions.

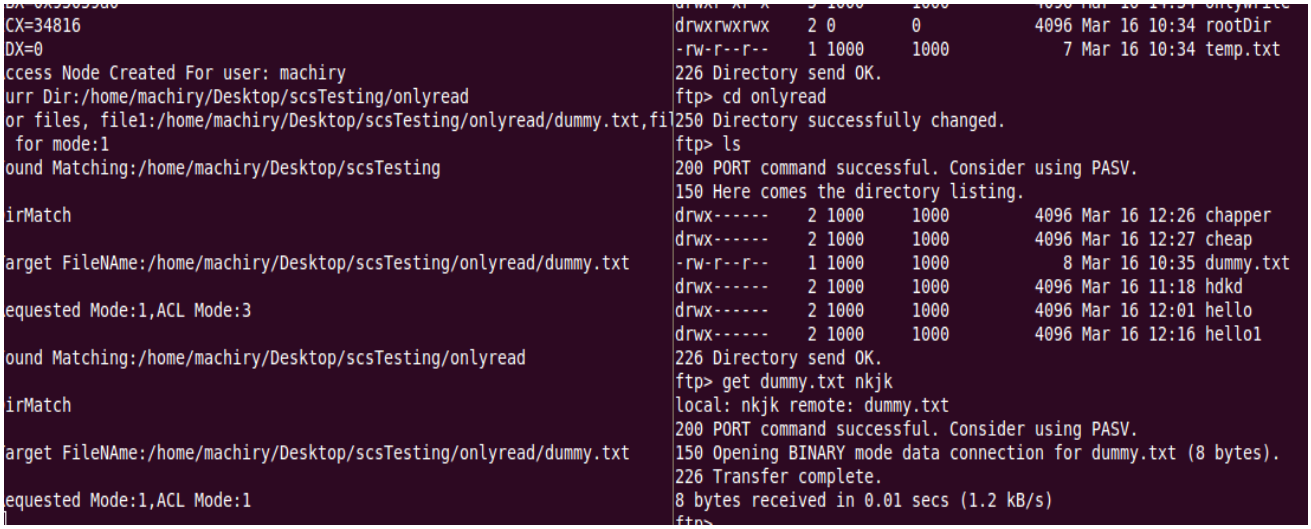
We tested our code by creating several permutations of users and groups accessing files in different modes and having different permissions for different kinds of access and our program was able to successfully grant/deny access in all the cases.

VI. Snap Shots:

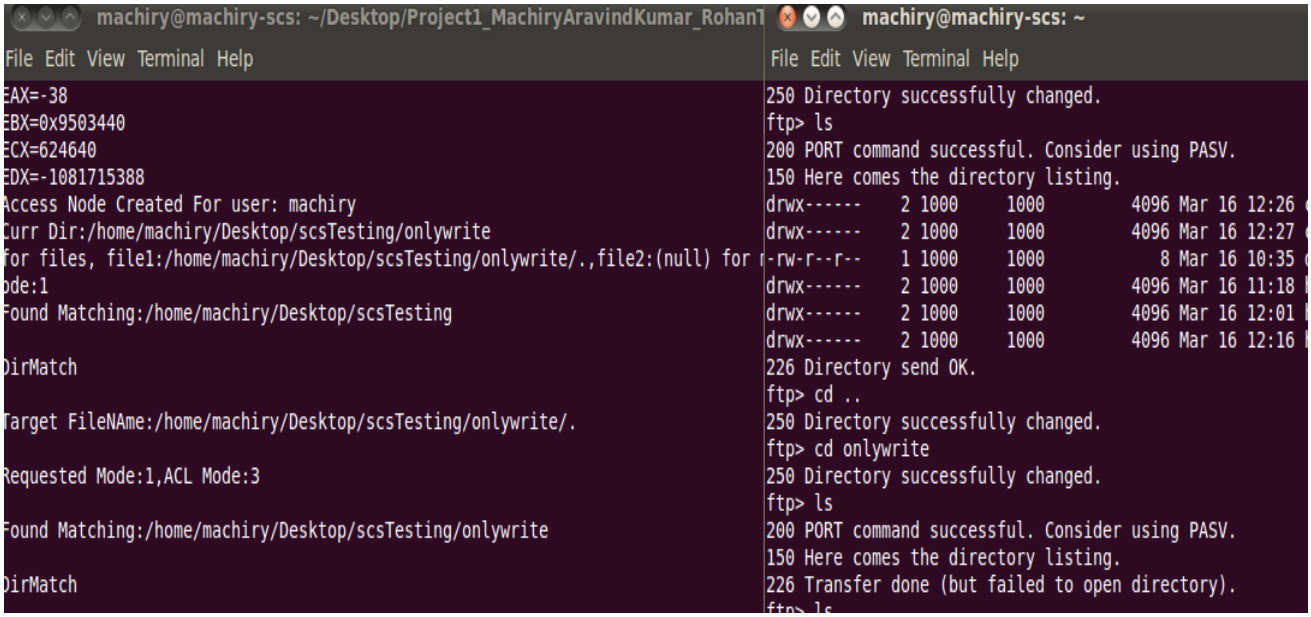
Flaw Finder Output:

```
machiry@machiry-scs:~/Desktop/Project1_MachiryAravindKumar_RohanTahiliani/project$ flawfinder src/
Flawfinder version 1.27, (C) 2001-2004 David A. Wheeler.
Number of dangerous functions in C/C++ ruleset: 160
Examining src/helper.c
Examining src/accessMode.h
Examining src/accessPolicy.h
Examining src/accessControl.c
Examining src/main.cpp
Examining src/logger.c
Examining src/helper.h
Examining src/accessPolicyParser.h
Examining src/commonHeaders.h
Examining src/accessControl.h
Examining src/sysCallStructs.h
Examining src/policyParser.c
Examining src/logger.h
src/accessControl.c:16:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
src/accessControl.c:19:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
src/accessControl.c:21:  [4] (buffer) strcat:
  Does not check for buffer overflows when concatenating to destination.
  Consider using strncat or strlcat (warning, strncat is easily misused).
src/accessControl.c:28:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
src/accessControl.c:31:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
```

Access Granted:



Access Denied:



Authentication Failed: