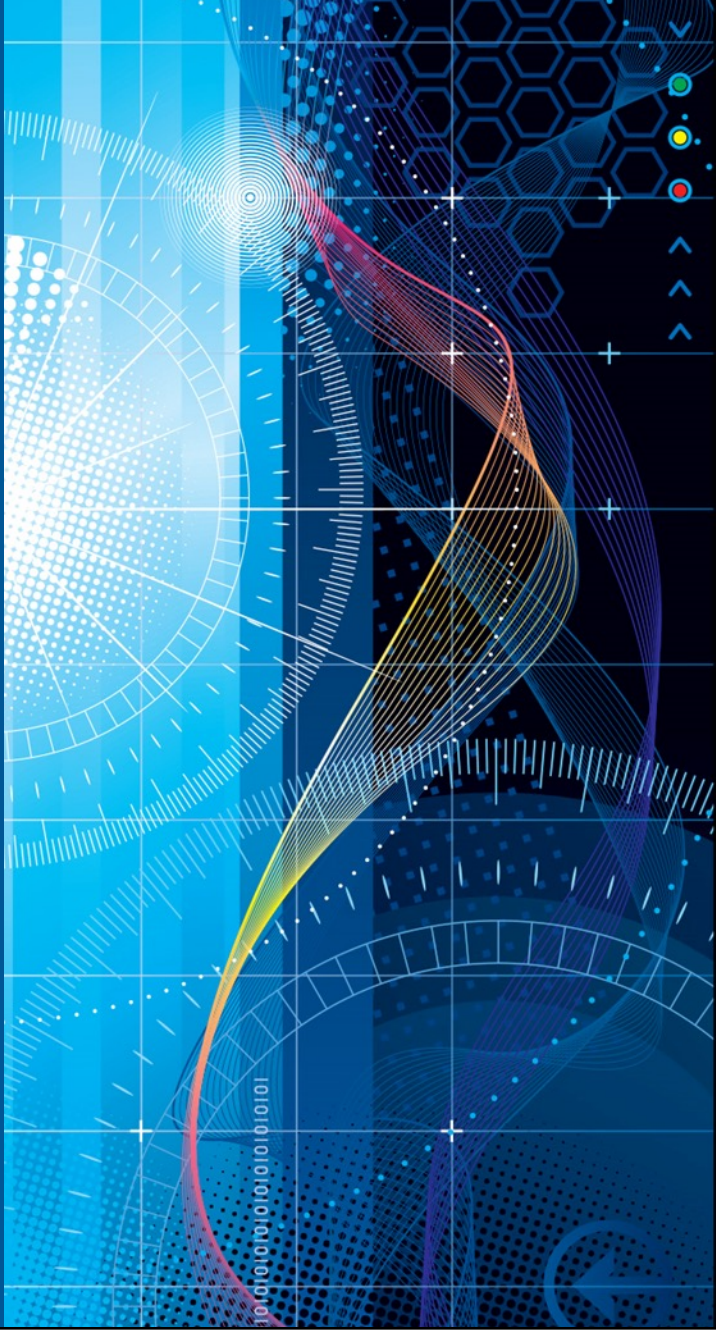
Secure Coding in C and C++

Exercise #6: File I/O Vulnerabilities



Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Notices

Sample Program

Caesar cipher decryption program

- Implements simple rotation cipher
- Takes input from files

In a real usage scenario, the decrypted file and the keys file must be kept secret from unauthorized users.

- Should only be usable by intended user
- The secret file can be used by anyone; it’s protected by encryption!

Usage

The program accepts a command line argument:

```
Usage: %s secret_file keys_file [output_file]
```

The `secret_file` argument specifies the name of the file containing the encrypted text.

The `keys_file` argument specifies the name of the file containing the corresponding “keys” to decrypt each line of the encrypted text.

- `keys_file` must live in home directory, or a subdirectory.
- `keys_file` can only be read with `root` privileges

The program also accepts an optional `output_file` argument.

- If `output_file` is not specified, the program prints the output to `stdout`.
- Otherwise `output_file` must be placed in home directory, or a subdirectory.

The Input Files

All of the files involved are just character files.

Each line contains the ciphertext and corresponding “key” (number of chars to rotate). For example:

Ciphertext	Key	Plaintext
`Lzak ak s lwkl`	8	`This is a test`

The lines are delimited by the EOL.

A working set of example files is included.

Exercise Tools

Find and fix any I/O security vulnerabilities in the `caesar` program:

- manual code reading
- compile and test

Use reference material.

- C99 standard
- POSIX standard
- man / help pages
- CERT Secure Coding standards
- *Secure Coding in C and C++*

Assume the program runs with root privileges

Exercises

Find and fix:

- I/O security vulnerabilities

(45 minutes)

Format String Vulnerability

In `usage()`:

Recall that `errmsg` is constructed from `getenv("USER")`

```
fprintf(stderr, errmsg);
```

Missing format specifier

File Validations

Input file and key file should exist and be readable.

- Accomplished by `fopen(filename, "r")`

Keys file should exist and live in home directory (or subdirectory)

- Accomplished using `realpath()`

Output file should not exist but be specified to live in home directory (or subdirectory)

- Accomplished using `realpath()`

Verifying Path lives in Home Directory

```
int in_homedir(char *const filename) {

  struct passwd *pwd = getpwuid(getuid());
  if (pwd == NULL) {
    return 0;
  }

  const size_t len = strlen( pwd->pw_dir);
  if (strncmp( filename, pwd->pw_dir, len) != 0) {
    return 0;
  }

  return 1;
}
```

Learns home directory from **`/etc/passwd`**

Returns true if path begins with home directory.

Requires **`filename`** to be absolute path and canonical!

Validating Canonical Path

```
FILE* canonicalize_and_open(char *const filename, char* mode) {
  char *realpath_res = NULL;
  char *canonical_filename = NULL;
  size_t path_size = (size_t)PATH_MAX;
  if (path_size > 0) {
    canonical_filename = malloc(path_size);
    if (canonical_filename == NULL) {
      errx(1, "Out of memory");
    }
    realpath_res = realpath(filename, canonical_filename);
  }
  char* errstr = NULL;
  if (realpath_res == NULL) {
    errstr = "Realpath failure";
    goto error;
  }
  if (!in_homedir(realpath_res)) {
    errstr = "Not in home directory";
    goto error;
  }
…
```

Perform home directory check on canonical path

Validating Canonical Path, cont.

```
FILE* canonicalize_and_open(char *const filename, char* mode)
{
…
  FILE* fd;
  if ((fd = fopen(realpath_res, mode)) == NULL ) {
    errstr = "File not found";
  }


error:
  free(canonical_filename);
  if (errstr != NULL) {
    errx(1, errstr);
  }
  return fd;
}
```

Clean up **`canonical_filename`** on any error

Putting It All Together

```
int main(int argc, char *argv[])
{
…
if ((infile = fopen(argv[1], "r")) == NULL)
      errx(1, "Cannot open input file.");

keyfile = canonicalize_and_open(argv[2], "r");

if (argc == 4) {
    outfile = canonicalize_and_open(argv[3], "w");
    oflag=1;
}
…
}
```

Use canonicalization to verify keys and output files

Questions