

HW 6

CS681 & CS392 Computer Security

Requirements and Design Document - Reviewed DUE 11/17/2005

1 Objective

In this homework you will **review and rework your own design document** of FEAU. Some comments I have given to you individually and some of the more generic ones are here.

2. Comments on Threats

- Don't limit threats to user related things only, like users passwd being compromised, user's changing things that he does not have permissions to. Things like the FRM being bypassed by FEAU are also threats.
- Identify threats at a component level also in addition to system level threats.
- You might not address all threats in the software that you develop. The threat FRM being bypassed by a process might be addressed by requirements like FRM should be brought up as soon as the system boots up. This might be outside you scope to implement but you should think about it and list it as a threat.
- Some of you have identified as few as 3 threats. How about identify at least 15 threats.

3. Comments on Assumptions

Some of you have listed assumptions like

Assumption: The system that assigns and checks ownership is secure and can't be exploited.

Comment: That "system" is what we are designing. We cannot think of it as an assumption. This is a requirement and we have to address this requirement. Assumptions are things like "The underlying encryption algorithm (for e.g. AES cannot be brute forced" since we are not going to handle what happens when the encryption algorithm is brute forced.

Assumption: Symmetric key generation is secure, and does not have predictable results.

Comment: This is a good observation that the AES keys should not be the same always. But that will be a requirement and how we address the requirement is that we seed the random number generator properly (with time for e.g.) so that the symmetric keys generated are "more random". This "how we address it" will be in the design.

Assumption: "Hashing passwords is secure process, and no clear text residue resides in memory once hashing algorithm finishes."

Comment: This is a good assumption. Since we use a hashing algorithm as a black-box we are justified in making such an assumption.

Assumption: The authentication process is a trusted process.

Comment: If you mean the login/password authentication of FEAU by this, then it is a reasonable assumption too. But it would be great if you had it as a requirement and then did something like hashing the FEAU and verifying the hash each time FEAU is started up. Ofcourse, the first hash has to be of a trusted process, which could be an assumption!

Assumption: FEAU is always online

Comment: This is a good assumption for you as a developer of FEAU & FRM. It would be a requirement for the guy who builds the underlying architecture, like the OS for e.g.

Assumptions: Root is a trusted user

Comment: This is a bad assumption. You should always think about what happens when the Root gets hacked.

4. Comments on Design

Sometimes the design is not very clear from your documents.

- Your design document should convey the design of the system completely and coherent and not design of individual bits and pieces. The best way to do this will be to develop Use-Case diagrams, but if that is too complicated you can use a flow diagram and specify who the caller of the function is.
- Work more on your architecture diagrams. Don't have source file names in your architecture.
- Have one overview diagram and have one diagram for each component (suggestion)
- You can also include a **state machine diagram** to make the design clearer. Infact, you can design the FEAU as a state machine. That would make your FEAU more object oriented. Implementing this in C might be a little difficult, but you can re-think your system in terms of objects. Keeping FRM stateless might be a better idea for simplicity, but you can think about it. (This comment is optional. Think about it. You don't have to do it.)
- Describe how you handle situations like when a user is deleted or when the user forgets his password. What happens to all the encrypted files.
- Address little things also in the doc. How do you plan to remove the plain text file after encryption? Say you are doing this for a unix system. Does rm remove the file in a way you cannot recover it back? Investigate. Look at what to do for a Windows system (This comment is optional. Think about it. You don't have to do it.)
- Don't lump too much functionality into the Reference monitor, like user and group management etc. FRM should be as small as possible since it is going to run with root privileges.
- Specify which module runs with what privileges.

5. Comments on Interfaces

- For interfaces of a component, make it clear which module is the user of that interface. For e.g. if you have a module called File Sharing which has a function called givePerms, you have to make it clear which module (if not which function in the module) is the user of this function.
- Some of your interfaces have been poorly designed. Re-design them.
- For the external functions, specify who has permissions to execute that function. Add a column if u need to. Also for interfaces in a module. For e.g. ls. If a user doesn't have read or write permissions on a file can he still do ls?
- Specify what your parameters do. For e.g. if you have a parameter called "flag" specify when it's true, when it's false etc.

6. Comments on Naming:

- If there is a threat T1 it will give rise to requirements R1_1, R1_2 etc. Threat T2 will give rise to requirements R2_1, R2_2 etc. This makes it easy to map threats to requirements.
- Name your functions properly. For e.g. it's not really clear what the role of isValid is. Evidently it is used to validate something. Say in comments what.

7. Comments on Layout

- Don't clump all requirements at one place and all threats at one place. Each threat gives rise to a set of requirements and a set of assumptions.

8. Comments on policy file:

- You can add a section describing the policy files.
- No one has explicitly addressed policy files in enough detail. Specify the format of the policy file, which module (preferably which function in the module) can access the policy file.
- Specify who has read permissions for the policy file and who has write permissions. How will you enforce it?
- How will you check that the policy file has not been changed by a malicious user.
- Think about permissions. You don't have to have only read, write and execute permissions like in UNIX. You can also have permissions to cp to mv etc. But having explicit permissions for each operation might be too cumbersome. Address this in your policies section.

9. Comments on logging

- Nobody has given enough thought to the logging process in the FRM. You can add a section describing it. You have to think about what will be logged, how the logs will be stored securely, who will have access to the logs and how will you know if the logs have been tampered with. Securing the logs is important aspect, since the first thing an attacker will do is tamper with the logs.

Finally: Don't worry. Whatever changes you make to your design document, you will not have to incorporate all of them in your code. So go ahead and come up with the best design you can.

Note: Design documents are better if you space out the time you spend on it. If you plan to spend 1 and a half hour on it for e.g. it will turn out better if you spend 30 minutes on 3 days rather than all the time on the same day.

10 What and how to Handin

Handin: You should submit assignment through my.poly drop box no later than 12 AM midnight on the due date.

You must zip all files related to the assignments and use the following convention to name the zip

file:

<First Name>_<Last Name>_<Lab#>.zip

Submit using the name of only one of the members in your group.

REMEMBER IF YOU DO NOT USE THIS NAMING CONVENTION YOUR ASSIGNMENT WILL NOT BE GRADED AND YOU WILL NOT RECEIVE ANY CREDIT.

PLEASE SUBMIT HOMEWORK ON TIME.