

Name:

Information Assurance: Homework 6 Answers and Comments

Due October 24, 2007. Late homeworks only accepted though October 26, so we may post the answer key in time to help students study for exam 2.

1. The Type Enforcement mechanism provided by SE Linux is very general. Can it encode the BLP rules of a specific security level lattice? Consider a set of security levels with clearances $H > M > L$ and categories P1 and P2. Identify the necessary types and domains and allow statements (as defined in page 26 of the class slides) to define a Type Enforcement system that would enforce the BLP simple security condition and *-property. Or identify the weaknesses in the Type Enforcement scheme that would prevent such a mapping.

One can map the BLP model for these labels, but it is very cumbersome. First, you create a type to represent each point in the lattice built from the three clearances and the two categories, e.g. H_P1_P2 , H_P1 , H_P2 , H , M_P1_P2 , M_P1 , M_P2 , M , L_P1_P2 , L_P1 , L_P2 , L

Then for each point in the lattice, define allow rules to enable a subject with that label to have appropriate read access to objects with “dominated labels” and write access to objects with labels that “dominate” the subject. This is domination with respect to the underlying lattice. The type enforcement mechanism has no concept of domination.

For example for a subject with label M_P1 , the rules would be

*# Subject dominates object and object dominates subject
allow M_P1 M_P1 :file {read, write};*

*# Subject dominates object
allow M_P1 M :file read;
allow M_P1 L_P1 :file read;
allow M_P1 L :file read;*

*# Object dominates subject
allow M_P1 M_P1_P2 :file append;
allow M_P1 L_P1 :file append;
allow M_P1 L_P1_P2 :file append;*

And so on and so for for each possible subject label.

2. Compare and contrast the Pitbull LX access control mechanism and the SE Linux Multiple Category Security (MCS) mechanism. Identify one way in which they are similar, and one way in which they differ.

Name:

These two mechanisms are similar in the structure of their security labels. Both are category only labels. They both apply the same operation to test for read and write.

Some differences include: MCS is discretionary and LX is mandatory. LX can apply to all objects in a system and MCS is focused on files.

3. Try LibSafe to detect a printf() buffer overflow error. You can access libsafe from <http://www.research.avayalabs.com/gcm/usa/en-us/initiatives/all/nsr.htm&Filter=ProjectTitle:Libsafe&Wrapper=LabsProjectDetails&View=LabsProjectDetails>. If you do not have administrative access on the target machine, do not run “make install”. Instead set the LD_PRELOAD environment variable to the location of the libsafe library, /home/shinrich/libsafe-2.0-16/src/libsafe.so.2.0.16, in my case. The man page for libsafe is posted at <http://www.cs.uiuc.edu/class/fa07/cs461/libsafe.8.html>. Show problem code snippet and report on results.

If you own the system you were testing on, you should have seen the libsafe output in /var/log/secure. If you did not own the system, you needed to compile the debug target. This would enable the stack to be printed to the console when a stack smash attempt was detected.

Sorry for the confusion on this with respect to EWS access. Sadly, I do not have EWS access right now. For possible future program using home works, the TSG guys have given everyone access to the CSIL machines.

4. You have been assigned the task of augmenting a virus scanner to detect the latest encrypted virus. You have an example of the virus in your isolation lab. Describe two techniques for identifying a general instance of the virus and describe one possible downside for each approach.

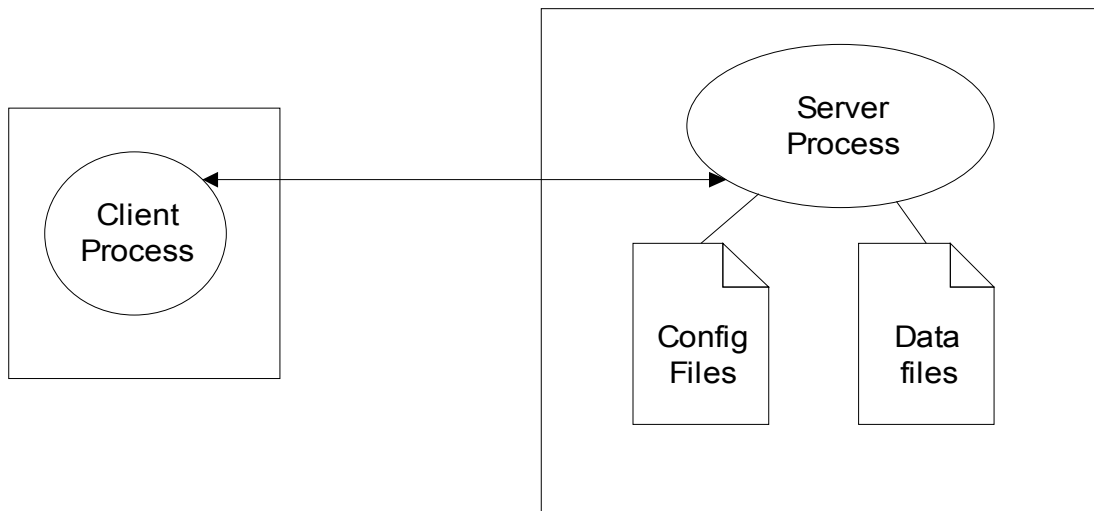
You could use pattern match techniques to find a likely decrypt loop. You could just use the presence of such a loop to signal a likely virus. However, this might catch legitimate code with an xor decrypt loop. You could use the key to decrypt the payload and compare with your example. You could use general virus detection techniques to see if the size of the binary is what you expect.

As an attacker, if I knew you had a signature for my decrypt loop, I'd rearrange by decrypt loop to not match your signature. Perhaps I would create a number of different versions of the same code or dynamically add some additional jumps (or other control elements) to break your specific signature.

If I knew that you were looking for file size, I'd hook the file size system calls to report the value you expect in the case of the corrupted binary.

Name:

5. Use the threat modeling technique discussed in class to analyze a rather high level system design of a banking client server application. A system diagram is shown below. The client process can run on a separate machine from the server. The operation of the server is controlled by configuration files stored on the server machine, and the persistent data is stored in data files on the same machine as the server machine. This is very early in the design so there is no real code for you to analyze. Identify assumptions that you are making about the system in your analysis.
- Identify two classes of system entry points.
 - Create one threat profile for a possible threat to the system.
 - Create a threat tree to illustrate that threat.



Banking system high level design

This was a very high level and thus vague question. In particular any number of threats could have been identified. The goal were was just to have you work through the process of thinking through threats and breaking them down into more fundamental steps.

- a) ID = 1 One entry point is the client network access to the server.
ID = 2 Another entry point is through the data files or configuration files

b) One possible threat profile

ID=34

Name = Attacker gains direct access to data files to adjust account levels

STRIDE = tampering, information disclosure

Entry point = 2 – Direct access through data files

Mitigated = don't know

Assets = Customer data

- c) One possible threat tree. As some of you noted, you can go on into very great detail. In this case, the threat tree is a starting point for discussion. As you divide the problem into smaller components you flesh out concrete attack possibilities.

Name:

As new vulnerabilities are discovered about your system, you can review your threat trees and see if the new vulnerability applies. As you discuss controls, you can see which controls will thwart specific attack paths in the tree. An control that thwarts more attacks or more high concern threats should be prioritized for implementation.

