Assignment 3 Solution Set

Thursday, March 21, 2019 11:30

A. Logging

Describe a security requirement that you think is appropriate for a file system, and for that requirement, describe the logging requirements of the following system calls:

1.) mkdir
2.) write

Solution:

mkdir: good logging should include: timestamp, if the call was successful, user information (UID, GID) at minimum. Answers describing when/how logging should happen for such a requirement were also accepted.

write: Some students interpreted the question as the write system call (writing to a file), and others used the write shell command (send messages to other users on the system). Points were given for both. Logging requirements were similar to mkdir, with different times logging should be invoked depending on the security requirement.

B. Logging and Audit

A successful attack was performed on a system at time t that gave the attacker full control of the system (including the power to modify the audit log from time t and beyond, but not before time t.) Given the log from time 0 to current such that 0 < t < current, would you still be able to identify that an attack happened on the system? Explain your answer. (Assume that all relevant events are logged in the audit log.)

Solution:

The key note here is the attacker gains control at time t. For an attacker to gain control of a system, he must perform some series of actions to gain control, and at minimum the final act must be some breach of our security model. These acts that break our security model will be logged (we assumed our logging system is a good one), and at minimum an investigator should be able to determine an attack took place, even if they are unable to know what happened during the attack.

C. Audit

Suppose when a successful compromise of a system is detected, the system administrator receives an email from the notifier. Describe a drawback of this mechanism? How should an appropriate user be notified?

Solution:

Points were given for 1) a reasonable drawback of the described system, and 2) providing an appropriate alternative.

D. Access Control Matrix

Assume, a computer system has 4 users: Kate, Alice, Bob, and Mike. Kate owns file Katefile, the other three people can only read the file. Mike and Kate can read and write file Alicefile (owner Alice). Only Bob can read and write the file Bobfile which he owns. Mike owns the file Mikefile and Bob and Kate can read and write the file Mikefile. Assume the owner of a file can execute it.

1. Create an access control matrix describing this system.
2. Suppose the owner of a file can add and remove permissions to that file. Mike removes Kate's permission and give permission to Alice to read and write Mikefile. Show the new access control matrix.
3. Suppose the commands to add or remove permissions can be written into a file, then that file is executed with the permissions of its owner. Does this change what operations are considered trusted? Explain.

Solution:

1)

	Katefile	*Alicefile*	*Bobfile*	*Mikefile*
Kate	*OX*	*RW*	**-**	*RW*
Alice	*R*	*OX*	-	-
Bob	*R*	-	*OXRW*	*RW*
Mike	*R*	*RW*	-	*OX*

2)

	Katefile	*Alicefile*	*Bobfile*	*Mikefile*
Kate	*OX*	*RW*	**-**	-
Alice	*R*	*OX*	-	*RW*
Bob	*R*	-	*OXRW*	*RW*
Mike	*R*	*RW*	-	*OX*

3) *If a user (user A) is granted write + execute permissions on a file owned by another user (user B), then user A gains indirect control permissions on that file (and all other files owned by user B). This changes the trust level of the write+execute operations.*

E. Access Control Matrix

Suppose we want to model a system with separation of duties, for example, that two people must agree to withdraw money from an account. One idea would be to create a file, give one person write access to the file, and another execute access. One person would write commands to the file to withdraw money, the other person would then execute the command.

1) If we assume the owner can grant any permissions to an object, we can show that this idea doesn't guarantee separation of duties. Show why.
2) How could we use the HRU access control matrix to model such separation of duties? Hint: instead of having subjects be able to execute primitive commands, they are only give a limited set of "programs" consisting of a sequence of primitive commands, as shown in slide 22 (page 7) of the 2/19 slides.

Solution:

1) Since the owner can grant arbitrary permissions, he can grant any subject he wants both write and

execute permissions, completely bypassing our scheme.

2) Instead of allowing the owner to arbitrarly change permissions for objects (files) they own, we give the owner access to three programs.

   Program 1 - Read Access: Can be used to grant or revoke read access.

   Program 2 - Write Access: Used to grant or revoke write access. If granting write access, program *always* removes execute access.

   Program 3 - Execute Access: Used to grant or revoke execute access. Similarly to Program 2, if we are granting execute access, it *always* revokes write access.

   These three programs ensure that no user could have both write and execute permissions on any file at the same time, and thus gives us our separation of duties.

F. Risk Analysis

The HRU access control matrix model, and attack graphs, both provide a way to formally model if security policies can be violated. What do we generally think of getting from the attack graph that we don't get in the HRU model?

Solution:

The main advantage of an attack graph is the ability to see the path of vulnerabilities used in an exploit. Using this we could find the least number of vulnerabilities to fix the greatest number of exploits on our system.