Patrick Austin

11/13/2017

CS 450 Homework 3

1. It is not true that H(x) ≠ H(x′) for all x and x′. Because the hash function maps a message of arbitrary bit length and the hash function’s output is a fixed n-bits long, H(x) will equal H(x′) for some x and x′.

For example, consider a hash function with an 8 bit output, i.e. 256 possible values for H(x). If 257 different x values are given as input to this hash function, there must be at least one collision where H(x) = H(x′) since there are only 256 possible values for H(x) but 257 values for x were given. To make this even more clear, consider a 2 bit hash output with 3 or more different values given as inputs.

2. Discretionary access control is an approach where an entity can grant access to a resource to other entities. For example, a file’s owner can grant write access to the file to other users. Those users can then grant write access to the file to other users, and so on. This chain of granted rights is often represented as an access matrix of users and file rights, which can be implemented using list data structures, either as an access control list or as a set of capability tickets.

Whereas a discretionary access control system defines access rights in terms of users, role-based access control defines access rights in terms of the roles of users in the system. For example, a ruled-based access control system for a university’s digital course content platform can create a set of rights for students, graders, faculty, administrators, advisors, etc. and simply assign individuals to one or multiple roles as need be.

There are several advantages to the rule-based approach. For one thing, a smaller access control matrix can often be used. Rather than maintaining access rights on a per-user basis, one can merely keep track of a user’s roles, and keep track of what a given role can do to a given object. For very large systems with many users, this is a much more efficient way to store access rights. The RBAC approach also makes it easy to add new users to the system: rather than having to create a new and bespoke set of appropriate access rights for each new user, one can simply assign them to the appropriate roles and be done with it. Similarly, one can easily add or remove roles from a user to modify the user’s access rights as the needs of the organization and system require it.

3. The original access control matrix is shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | File 1 | File 2 | File 3 | File 4 |
| User A | Own, R, W |  | Own, R, W |  |
| User B | R | Own, R, W | W | R |
| User C | R, W | R |  | Own, R, W |

4. To create the new diagram of access right dependencies, we follow the convention stated in the textbook: when a user A revokes an access right, any cascaded access right is also revoked, unless that access right would exist even if the original grant from A had never occurred. Following that convention for B revoking C’s access, we get the following diagram:

