CS5430 – System Security Spring 2018

Assignment 4

Due: April 23, 2018

Problem 1: Capabilities

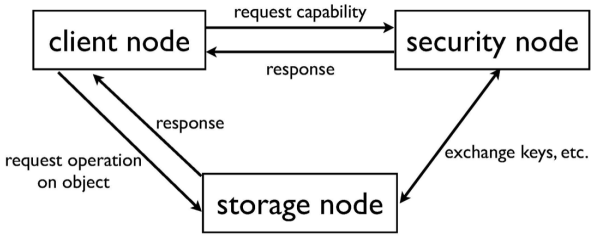
You have been engaged as a security consultant by Yangtze,[1] a new company providing cloud storage. Yangtze's new system is named Remote Repository, or R2 for short. With R2, Yangtze's engineers seek to build an ultra-high performance cloud storage system. They've solved most of the problems, but they need your help with the access control subsystem.

Yangtze built a prototype of R2 that uses access control lists. They've encountered a serious problem, though: every request that a client makes to read from or write to an object in storage has to be authenticated, the client has to be mapped to a subject, and the subject's entry in the ACL for the object has to be consulted. All that work is slowing down the system, keeping it from achieving Yangtze's performance goals.

Luckily, you studied capabilities in CS 5430. You know that with an access control subsystem based on capabilities, the storage system would need to do very little work, because the client would simply present the capability along with its request. The storage system would need only verify that the capability permits the request. Also, Yangtze is excited about the possibility of subjects delegating access rights without ever having to contact R2 at all, because this would further enhance performance.

But there is one big problem: you've read about how to implement capabilities with asymmetric cryptography, digital signatures in particular, but that kind of crypto is too slow for use in R2. You're going to have to find a way to implement capabilities with symmetric cryptography.

So far, you've invented the following architecture for the system:



- The client node is used to access R2.
- The security node authenticates clients and issues capabilities.
- The storage node verifies capabilities when they are used to access objects.

You've already taken care of the authentication subsystem—it doesn't play much, if any, role in the work you're doing now. Furthermore, you've already arranged that the security node and storage node can share an arbitrary number of symmetric keys—you don't need to concern yourself with how to accomplish that key distribution. However, generating and distributing keys is somewhat expensive, so Yangtze insists that you keep the number of keys used to a minimum. Finally, you can assume that all communication channels between client nodes and server (i.e., security and storage) nodes are secured with SSL in unilateral authentication mode: the client authenticates the identity of the server,

[1] The Yangtze is the next-longest river in the world after the Amazon.

all communication is encrypted to protect confidentiality, and replay of messages sent over the SSL channel is detected.

One more thing: Yangtze has provided you with an implementation of globally unique identifiers (GUIDs) for objects, so that every object in the system has its own unique 128-bit identifier.

Your remaining work is to figure out how to handle the following concerns:

- How R2 will grant access to clients by issuing capabilities when they are requested?
- How R2 will determine access by deciding whether a subject may read or write an object?
- How R2 will enable delegation of access between subjects?
- How R2 will enable revocation of access to objects?

Taking into account all the constraints and goals above, you now need to produce a design for R2's access control subsystem. You will want to carefully specify what capabilities are: what fields they contain, how to interpret those fields, etc. You'll also want to explain in detail how each of the above concerns will be implemented in R2. If there are any cryptographic protocols involved, you need to write those down using proper notation, and explain each step. Finally, explain why you introduce each symmetric key that is used in your design, and explain why you've used the minimum of number of keys necessary.

Problem 2: Multi-Level Security

Consider a new scheme for implementing MLS confidentiality, where labels on files and programs can be changed according to the following rules.

- At any time, the label $L(F)$ on a file F (i) can be increased or (ii) can be decreased to the largest label on any item that has been written so far to that file.
- At any time, the label $L(Pgm)$ on any program Pgm (i) can be increased or (ii) can be decreased to the largest label on any item that has been read so far by that program.

Moreover, assume that

- If $L(Pgm) > L(F)$then a write to file F by program Pgm—that is, a "write-down"—is implemented as a "no-op" (but does not cause program execution to be blocked or terminated).
- If $L(Pgm) < L(F)$ then a read to file F by program Pgm—that is, a "read-up"—returns "file unavail" as if that is the contents of the file.

Is there an environment where it is possible for a program Pgm to learn whether the contents of a file F satisfy some given predicate, even though $L(Pgm) < L(F)$ holds? (We define environment to mean: some set of files and other executing programs.) If so, describe the environment and the attack; if not, give an argument that explains why the information cannot be learned by Pgm.

Problem 3: Reclassification

Sometimes the secrecy of individual data are less sensitive than their aggregate. For example:

- The budgets of individual departments of a company may not reveal much information. But collectively, they reveal where the company is concentrating its resources, and thus telegraph
- In the movie Mission: Impossible (1996), the recovery of a NOC (non-official cover) list is a focus of Agent Ethan Hunt. One half of the list contains the code names of secret agents, and the other half contains the agents' real names. Each half individually reveals sensitive information, and their combination reveals even more information.

Aggregation is particularly relevant in the context of databases. For the purpose of this problem, suppose that a database comprises a number of datasets. (A dataset might be a table or a view.) Further, suppose that each dataset is assigned a sensitivity label such as Unclassified, Secret, or Top Secret. (We ignore compartments in this problem.) Then it might be the case that datasets A and B are both Unclassified, but that their aggregation is Secret. To model this, let the function $L(R)$, where R is a set of datasets—for example, $R = A, B$—denote the sensitivity of the aggregation of all the datasets in R. As healthiness conditions on L, we require that:

- For all A, $L(A)$ = sensitivity of A.
- If $R \subseteq R'$ then $L(R) \leq L(R')$.

Our goal in this problem is to develop a MAC model for this scenario. Suppose that an object is a document containing information derived from the database—e.g., the result of queries on datasets. A subject, as usual, is a process executing on behalf of a user. An entity is either a subject or an object.

Part A

Construct your own real-world example, using the database model above, of aggregate data that are more sensitive than their constituents. (However, the particular examples above and examples from lectures are off-limits. Be original. If you need inspiration, begin by supposing that one of the datasets is a set of photographs.) Your example should include at least three datasets. Identify what $L(R)$ is for each possible subset R of your datasets.

Part B

Suppose that each object (and subject) is labeled with its sensitivity (or clearance). We could then attempt to employ the Bell and LaPadula security conditions ("no read up, no write down"). However, we claim that these conditions are insufficient to guarantee the following policy:

P1: An object never contains information whose sensitivity is higher than the object's label.

Using your example database from Part A, prove this claim by exhibiting a series of read and write operations that effect such an information flow. You may freely invent entities and their labels.

Part C

Instead of sensitivity, suppose that each entity is labeled with a set of datasets. Give new conditions for reading and writing. Your conditions should be general—not specific to the particular example you gave in Part A—and should guarantee the following policy:

P2: If X is labelled with R, then the information in datasets R should be allowed to flow to X, and information from datasets other than those in R should not be allowed to flow to X.

Problem 4: Noninterference

The security policy noninterference for confidentiality can be informally stated as, "Changes in secret inputs do not cause changes in public outputs." Do the following programs satisfy this policy? Argue why or why not. Your interpretation of this informal condition may affect your answer, so the clarity and precision of your rationale is important. If your answer depends on some assumptions (e.g., you assume that covert channels like timing and termination are or are not observable) you should clearly document your assumptions. In all cases, the program P has two input strings—H_I (high in) and L_I (low in)—and two output strings—H_O (high out) and L_O (low out).

(A) P outputs $H_O = H_I || L_I$ and $L_O = L_I$, where $||$ denotes string concatenation.

(B) P outputs $L_O = L_I$ if H_I is even and $L_O = L_I || L_I$ if H_I is odd.

(C) Assume H_I is always a 32-bit binary string. P generates a uniformly random number 32-bit binary string k, then outputs $L_O = H_I \oplus k$.

(D) Assume L_I is an RSA public key. P outputs $L_O = \text{Enc}(H_I; L_I)$.

Problem 5: Information Flow for Integrity

In class, we studied information flow control for confidentiality. We saw a simple lattice with two labels, L and H, which were interpreted as policies constraining the flow of information: data tagged L were public, whereas data tagged H were secret. In this problem, we examine information flow control for integrity. So instead of secrecy, we turn our attention to the trustedness of data: some data are trusted, others are untrusted. For brevity, let's give those two policies names: P_1: trusted, P_2: untrusted.

As in class, we use a lattice to model information flow: Let Λ be the set $\{L, H\}$ of labels. Let $\sqsubseteq$ be a relation on labels that characterizes when information may flow, where $L \sqsubseteq L, L \sqsubseteq H, H \sqsubseteq H$. Let $\sqcup$ be an operation on labels that characterizes the label that results when data are combined, where $L \sqcup L = L, L \sqcup H = H, H \sqcup H = H$. But we now want to use labels L and H to represent integrity policies, not confidentiality policies.

(A) Which label, L or H, represents policy P_1, and which represents policy P_2? Explain why your answer is in accordance with the fact that $L \sqsubseteq H$. Also explain why your answer in accordance with the fact that $L \sqcup H = H$.

(B) Here is a definition of noninterference for integrity:

$$\forall M_1, M_2, M_1 =_L M_2 \Rightarrow C(M_1) =_L C(M_2).$$

The intended intuition behind that definition is that changes to untrusted variables should not cause changes to trusted variables. You'll notice that it is, in fact, syntactically the same definition of noninterference that we gave for confidentiality.

The type system we discussed in lecture soundly but incompletely enforces that definition of noninterference for integrity—that is, the type system rejects insecure programs but also rejects some secure programs. Give two example programs that demonstrate this incompleteness. The first example should be an assignment statement. The second example should be an if statement whose guard contains at least one variable. For both examples, provide a mapping Γ from variables to labels. Also, if your examples include any constants, give the label for those constants.

(C) With confidentiality, some functions could be treated as changing the secrecy of data—for example, encrypting a secret plaintext could be treated as producing a non-secret ciphertext. Give an example of a function that could be treated as increasing the integrity of data. And give an example of a function that decreases the integrity of data.