Instructors: Matt Fredrikson, Jean Yang TA: Samuel Yeom

Due date: March 7 at 11:59pm

Assignment 3

Last updated on February 27

To give some of the class a chance to polish their servers from previous assignments, this homework consists of only written exercises. There is also an optional extra credit proof.

This homework is based on the following scenario. Suppose that members of the class can implement extension modules to the server scripting language. We want to support many different extensions, but we also want to make sure that the extensions are safe. Therefore, it is important to automatically determine which extensions we can trust, and to limit the damage of programming errors made by other classmates.

You will first work through some theoretical exercises about access control logic and proof-carrying code, and then you will design a system that is secure in this scenario.

1 Access control logic (14 points)

Recall the access control logic presented in lecture:

$$
\begin{array}{rcl}
p & ::= & \textbf{key}(s) \mid \mathsf{identifier} \mid p.s \\
\phi & ::= & \textbf{action}(s) \mid p\ \textbf{says}\ \phi \mid p\ \textbf{speaksfor}\ p \mid s\ \textbf{signed}\ \phi \mid \textbf{delegates}(p, p, s) \mid \phi \to \phi \mid \phi \land \phi
\end{array}
$$

$$
\textsc{Says-I1}\ \frac{s\ \textbf{signed}\ \phi}{\textbf{key}(s)\ \textbf{says}\ \phi}
\qquad
\textsc{Says-I2}\ \frac{\phi}{p\ \textbf{says}\ \phi}
\qquad
\textsc{Says-I3}\ \frac{p\ \textbf{says}\ (p.s\ \textbf{says}\ \phi)}{p.s\ \textbf{says}\ \phi}
\qquad
\textsc{Says-Impl}\ \frac{p\ \textbf{says}\ (\phi_1 \to \phi_2) \qquad p\ \textbf{says}\ \phi_1}{p\ \textbf{says}\ \phi_2}
$$

$$
\textsc{Speaksfor-E1}\ \frac{p_1\ \textbf{says}\ (p_2\ \textbf{speaksfor}\ p_1) \qquad p_2\ \textbf{says}\ \phi}{p_1\ \textbf{says}\ \phi}
\qquad
\textsc{Speaksfor-E2}\ \frac{p_1\ \textbf{says}\ (p_2\ \textbf{speaksfor}\ p_1.s) \qquad p_2\ \textbf{says}\ \phi}{p_1.s\ \textbf{says}\ \phi}
$$

$$
\textsc{Delegates-E}\ \frac{p_1\ \textbf{says}\ \textbf{delegates}(p_1, p_2, s) \qquad p_2\ \textbf{says}\ \textbf{action}(s)}{p_1\ \textbf{says}\ \textbf{action}(s)}
$$

In this problem, you will prove some simple properties about the authentication scheme. Each member of the class is a principal, and the course staff is the certificate authority CA. The course staff vouches for each member of the class. Different members of the class can choose to delegate authority to other members of the class, based on who they decide to trust for extensions.

Suppose `Steve`, a member of the class, wants to run his extension. We represent this as

$$K_{\texttt{Steve}}\ \textbf{signed}\ \textbf{action}(\texttt{run}). \tag{1}$$

In order to convince principal p to run the extension, we must be able to prove that

$$p.\texttt{trusted}\ \textbf{says}\ \textbf{action}(\texttt{run}). \tag{2}$$

(a) (2 points) The course staff needs to vouch for Steve's key, so that other students can trust that statements signed by Steve's key represent his statements. To do so, it will issue a certificate of the form:

$$K_{\mathsf{CA}}\ \textbf{signed}\ (\_\_\_\_\_\ \textbf{speaksfor}\ \_\_\_\_\_).$$

Fill in the blanks in the above statement.

(b) (2 points) Even with the voucher from CA, we still cannot prove Statement 2 because we do not have any policies about p.trusted. What policy does p need to complete the proof?

(c) (4 points) Use the statements from parts (a) and (b), along with Statement 1, to complete the proof. For each step of the proof, state the inference rule used.

(d) (1 point) Now we consider another principal called Hub, who simply says statements rather than signing them. This is bad practice because anyone can impersonate Hub, but we will ignore that issue for now. Hub.students is a group of all CMU students. What does Hub say to represent the fact that Steve is a member of Hub.students?

(e) (1 point) p wants a policy that says any CMU student is trustworthy. What statement should we add?

(f) (4 points) Use the statements from parts (d) and (e), along with Statement 1, to prove Statement 2. For each step of the proof, state the inference rule used.

Solution

(a)

(b)

(c)

(d)

(e)

(f)

2 Proof-carrying code (20 points)

The certificate authorities help with not running code from untrusted sources, but there may be some clumsy programmers at CMU, in your class, or even among your trusted friends. With type-safe languages like OCaml, this is less of a problem, but imagine a scenario where a classmate decides to write optimized C code, thus introducing all kinds of potential memory errors.

In this problem, you will work with proof-carrying code to make sure that there are no array out-of-bounds or null pointer access issues in compiled extension modules.

This problem is based on 5.1-5.2 from the Pierce reading, included with the assignment. We will be reviewing the typing rules on Tuesday, February 28.

(a) (6 points) Add a new array type constructor to the safety policy and write the proof rules for its usage. An array is represented as a pointer to a memory area that contains the number of elements in the array in the first word and then the array elements in order. Consider the case where each element is a word type.

(b) (2 points) Discuss how the proof of soundness would need to change to accommodate this new rule.

(c) (10 points) Provide the proofs for the SET, THIS, SEL, and UPD cases of Safety Policy soundness.

(d) **Extra credit (8 points):** Provide a proof of soundness of the new rules for the Safety Policy. (See the "Soundness of the Safety Policy" section of the chapter.)

Solution

(a)

(b)

(c)

3 Designing a secure system (18 points)

Now you will consider the trade-offs of the following methods for designing a secure extension framework for the server:

- Proof-carrying authentication using access control logic (Problem 1)
- Proof-carrying code (Problem 2)
- Software fault isolation (from lecture)

(a) (6 points) For each of the above methods, state what needs to be implemented on the server and what the authors of the extension modules need to do.

(b) (6 points) Discuss how each of the above methods contributes to the following security principles:

1. Complete mediation
2. Smallest trusted computing base
3. Least privilege

(c) (6 points) If you had to pick two of the three methods for the server, which would you choose, and why? Would your answer change if you were building an extension framework for Chrome?

Solution

(a)

(b)

(c)