Computer Security

Prof. Dr.-Ing. Volker Roth
Freie Universität Berlin

April 24, 2012

Question 1: State transitions

Given the following state transition model based system:
System policy "A subject U may access an object O if it holds that $U.sclass \geq O.oclass$"
There are currently three users (U_1, U_2, U_3) on the system working on three different objects (O_1, O_2, O_3). All documents are in *rw* mode. U_1, U_2, O_1 and O_2 are at security level 1. U_3 and O_3 are at security level 2. Assume that the users perform the following actions

1. U_1 sets O_1 to read-only r
2. U_3 lowers security level of O_3 to 1
3. U_2 creates a new object O_4 with rw permissions and copies the content of O_3 to O_4
4. U_3 raises security level of O_1 to 2
5. U_3 raises security level of O_3 to 2

- Fill out the following tables representing the final state after the actions described above took place.

Table 1: Access control matrix

	O_1	O_2	O_3			U_1	U_2	U_3
U_1						-		
U_2							-	
U_3								-

Table 2: Security levels

O_1	O_2	O_3		

- Answer the following questions with one sentence only
  1. Can U_3 write O_1 after step one?
  2. Can U_2 read O_1 after step five?
  3. Can U_1 access the content of O_3 after step five?
  4. Can U_3 change the permissions on O_4 after step four?

- Write a short (one sentence) additional policy to prevent U_1 and U_2 from accessing the content of O_3.
- Based on your previous observations would you judge the security policy to be sufficient for a secure system?

Question 2: Rainbow tables

You are asked to recover a password from the hash "xbdz". You are provided with the definition of the used hash function and a previously generated rainbow table.

Hash function (h)
The given hash function only takes lowercase characters [a-z] as input.It takes the input and translates every character to it's corresponding number (e.g. c=2). It then sums up the values pairwise and translates the result back to a corresponding character.
Addition of two characters is defined as adding the corresponding integer values (a=0, b=1, ..., z=25) and using the character corresponding to the resulting integer as the final result.
Let I be the input to the hash function.
if $|I| \ mod \ 2 = 1$ append "a" to I
Let H be the resulting hash.
$H_i = (I_i + I_{i+1}) \ mod \ 26$

Example: $h(uaregood) = uvur : u + a, r + e, g + o, o + d \rightarrow 20 + 0, 17 + 4, 6 + 14, 14 + 3 \rightarrow 20, 21, 20, 17 \rightarrow uvur$

Regeneration function (r)
The regeneration function doubles the length of the input. It inverts the input and adds $b = 1$ to every character. Afterwards it concatenates the input with the calculated extension.
Let R be the Regenerated text.
Let Q be th inversion of H
$R_i = H_i$ for $i \leq 0 < 5$
$R_i = (Q_i + \text{"}b\text{"}) \ mod \ 26$ for $i \geq 4$

Example: $r(uvur) = uvursvwv : Q = ruvu \rightarrow u, v, u, r, r+b, u+b, v+b, u+b \rightarrow uvursvwv$

Rainbow table

initial password	end of chain
uaregood	uyaw
helloWor	osuq
password	mqso
geheimni	cgie
iamyourg	cgie
mondayss	imok
octoberh	uyaw
ucouldnt	imok
shouldud	ycea
starwars	koqm

Use the given definitions of the hash function and the regeneration function together with the given rainbow table to answer the following questions.

- Name a valid password?
- In which row did you find the password?
- How many steps did it take you to find the correct row?
- How many steps did you need overall?
- The given definitions lack a central feature of rainbow tables. Explain in one sentence what was left out.

Question 3: Optional - Basic assembler

- Convert $0xBEEF$ to decimal and from decimal to binary by hand.
- What are xIP, xBP and xSP ($x \in \{E, R\}$)? Where are they pointing at?
- Read http://software.intel.com/en-us/articles/introduction-to-x64-assembly/
- What is the target architecture of the code below?
- Compile the following code with gcc and run it. (The code was tested on Mac OS X and should run on Linux. It's pretty uncertain that it will even compile under windows.)
- Shortly explain every command in the following code

```
1  .cstring
2  _hello: .asciz "Hello, world\n"
3  .text
4  .globl _main
5  _main:
6  sub $8, %rsp
7  lea _hello(%rip), %rdi
8  call _printf
9  add $8, %rsp
10 ret
```