Computer Security

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

April 30, 2012

Question 1: Undecidability of the general halting problem

Consider the representation of a Turing machine as a protection system as described in section 4.7.2 of *Robling Denning, D. E. 1982, Cryptography and Data Security*.

- Specify the missing commands for the head moving right: $\delta(q, X) = (p, Y, R)$
- Given the access matrix below, show the matrix that results after the following two moves:
  1. $\delta(q, C) = (p, D, R)$
  2. $\delta(p, D) = (s, E, R)$

A	B	$\Downarrow$ C	D	ϕ	...

A	*own*		
	B	*own*	
		C q	*own*
			D END

Question 2: Real world access control matrix

Given the following file listing from a Linux file system

```
drwxr-xr-x  alice users .
drwxr-xr-x  alice users ..
-rwxr--r--  alice f     A
-rwxrw----  alice bfct  B
-rwxr-----  frank c     C
-rw-r-----  bob   ct    D
-r-xr-x---  tim   at    E
-r-----r--  carl  c     F
```

Assume that the following groups exist

- f - frank
- bfct - bob, frank, carl, tim
- c - carl
- ct - carl, tim
- at - alice, tim
- other - System predefined group

We define the following set of rights:

- own - The subject S owns the object O if $own \in PS, O$
- $read$ - The subject S may read the object O if $read \in P[S, O]$
- $write$ - The subject S may write the object O if $wrie \in P[S, O]$
- $exec$ - The subject S may execute the object O if $exec \in P[S, O]$

In addition we define a right i which allows a subject to use the rights of another subject "indirect". Hence if Carl has the right i on c he can in addition to his direct rights act with the rights of c, in this case: $read$ C.
More formal: Let R be an arbitrary right. If $R \in P[S_1, O]$ and $i \in P[S_2, S_1]$ it holds for all operations of S_2 that $R \in P[S_2, O]$.

Fill out the following access control matrix representing the access rights given in the file listing.

	A	B	C	D	E	F	f	bfct	c	ct	at	other
Alice												i
Bob												i
Frank												i
Carl									i			i
Tim												i
other												-
f												
bfct												
c			$read$									
ct												
at												

Question 3: Optional - Assembler addressing

- Assume $\%ebp = 0x80$, $\%eax = 0x8$. Calculate the effective adresses of the following displacements
  - -4(%eax, %ebp)
  - (,%eax,8)
  - 128(%ebp)
- Carefully look at the assembler dumps A and B given below.
  - Compare what the two programs do
  - What is the difference between the two programs
  - Give an equivalent C program
- Compile your C program with GCC
- Figure out how to disassemble your program with GDB
- Compare your dump with the given ones

Listing 1: A

```
0x0000000100000ef0 <main+0>: push %rbp
0x0000000100000ef1 <main+1>: mov %rsp,%rbp
0x0000000100000ef4 <main+4>: sub $0x10,%rsp
0x0000000100000ef8 <main+8>: lea 0x51(%rip),%rax # 0x100000f50
0x0000000100000eff <main+15>: mov %rax,%rdi
0x0000000100000f02 <main+18>: callq 0x100000f24 <dyld_stub_puts>
0x0000000100000f07 <main+23>: movl $0x0,-0x8(%rbp)
0x0000000100000f0e <main+30>: mov -0x8(%rbp),%eax
0x0000000100000f11 <main+33>: mov %eax,-0x4(%rbp)
0x0000000100000f14 <main+36>: mov -0x4(%rbp),%eax
0x0000000100000f17 <main+39>: add $0x10,%rsp
0x0000000100000f1b <main+43>: pop %rbp
0x0000000100000f1c <main+44>: retq
```

Listing 2: B

```
0x0000000100000f10 <main+0>: push %rbp
0x0000000100000f11 <main+1>: mov %rsp,%rbp
0x0000000100000f14 <main+4>: lea 0x39(%rip),%rdi # 0x100000f54
0x0000000100000f1b <main+11>: callq 0x100000f2a <dyld_stub_puts>
0x0000000100000f20 <main+16>: xor %eax,%eax
0x0000000100000f22 <main+18>: pop %rbp
0x0000000100000f23 <main+19>: retq
```