Computer Security

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

June 25, 2012

Please note that your submission will not be accepted without the properly signed Academic Integrity agreement.

Academic Integrity

We each certify that this submission is our own original work and that we have due acknowledged any work of others.

_____

Date, signature, name in block letters

_____

Date, signature, name in block letters

_____

Date, signature, name in block letters

Question 1: Building shellcode

It's highly recommended that you use some kind of virtualization for the following task!

Write a shellcode (in assembly by hand) that runs "/bin/sh" via a call to execve.

1. The shellcode shall be position independent (no hardcoded addresses)
2. Your shellcode must be minimal, if you don't really need a certain command leave it out
3. Explain the idea behind your shellcode
4. Explain every line of your assembly source
5. Hand in the hexdump of your shellcode along with a mapping of every byte to the corresponding command or data in the assembly source

Question 2: Packer and unpacker

It's highly recommended that you use some kind of virtualization for the following task!

As your shellcode may contain NULL-bytes you need a packer which removes NULL-bytes from your shellcode and an unpacker which restores these bytes. An easy way to achieve this is to XOR the shellcode with a byte such that neither a NULL-byte is left nor one is newly created.

1. Write a packer in C
   - The packer must take the hexdump of a shellcode as input
   - The packer has to find an usable byte for masking automatically
2. Write an unpacker in assembly and combine it with your shellcode
   - The unpacker shall be position independent (no hardcoded addresses)
   - Your unpacker must be minimal, if you don't really need a certain command leave it out
   - Explain the idea behind your unpacker
   - Explain every line of your assembly source

Hints

- You may test your shellcode if you use it in your program from the previous task
- As the shellcode "lives" within another program it can't have an own .data section. You have to find a workaround for this limitation.
- If you have trouble with ASLR you may disable it with "echo 0 > /proc/sys/kernel/randomize_va_space"
- The following compiler options may help you if you had problems with the previous task
  - fno-stack-protector
  - D_FORTIFY_SOURCE=0
  - z execstack