Evaluation activity.

Fill-in your data

N. USP	**Name**	**e-mail**
10892680	Luiz Fernando S. E. dos Santos	lf.santos@usp.br
10425396	Luca Gomes Urssi	lucaurssi@usp.br
Repository:	https://github.com/courselab/syssoft-xx_h4ck34d0_xx	

Directions.

- Answer the following questions as if you were explaining to a student taking they first undergraduate course on operating systems: be complete, yet concise, and provide all the explanations needed to understand the concepts and techniques involved fully.

- Unless otherwise specified, all examples assume an execution environment based on GNU/Linux operating system, Bash, GCC toolchain, GNU Make, and GNU Binutils.

- The list can be solved by teams of up to 4 participants.

- You may answer in English or Portuguese, it is indifferent.

- All source-code excerpts mentioned in the exercises can be found in the repository of examples at *http://gitlab.com/monaco/syseg*, and the path, when indicated, is relative to the root of the project tree.

1) Consult the file **`eg/try/hack01/README`** and proceed as indicated.

a) Explain how you solved the challenge.

R: We circumvented the access restriction by pre-loading a library(libfakeID) which contains the function "authorize", by doing so the new function returns access after any login input. To find which library to pre-load we used the command "objdump -d" on both docrypt and libauth.so, which showed us the mentioned library.
To get the encryption key we used "objdump -D" on encrypt finding that the key was a read only element, knowing that we listed all strings from the encrypt

binary and found the line which most probably was the key, after testing we made it show o console.

b) Create a subdirectory `hack01` in your project and upload the relevant files along with a Makefile script with instructions.

Tips.

- You should look for vulnerabilities (often resulting for misguided programming practices).
- Remember that you cannot modify the programs, or the changes will be detected by the security mechanism. Instead, you may consider some ideas discussed in the slides "8 Run (I)" can be useful.

R:

2) Consult the file `eg/rop/eg-09.c` and proceed as follows.

a) Read the description of the source code in the file `eg/rop/eg-09.c/README`. and follow the indicated steps to crack the program.

R: Os passos foram seguidos, resultando na execução do shellcode que imprime a string “Hacked!”, conforme o esperado.

b) Explain the illustrated return-oriented exploit.

R: O exploit de injeção de shellcode funciona por meio de um ataque de estouro de buffer. Nesse tipo de ataque, ocorre a sobrescrita do endereço de retorno de uma função, redirecionando o fluxo de execução do programa para um código malicioso.
Primeiro, o invasor aproveita uma vulnerabilidade no programa-alvo, que não realiza uma verificação adequada dos limites do buffer utilizado para armazenar a entrada do usuário. Isso permite que o invasor insira mais dados do que o buffer pode acomodar, causando um estouro e sobrescrevendo dados adjacentes na memória.
Em seguida, o invasor prepara um arquivo de entrada que será utilizado como entrada padrão do programa. Esse arquivo contém o shellcode, que é o código de máquina representando o programa malicioso a ser executado. Além disso, o invasor inclui um endereço de retorno manipulado, apontando para o local onde o shellcode está inserido no arquivo.
Para aumentar a probabilidade de sucesso, é comum utilizar uma sequência de instruções NOP no início do arquivo de entrada, conhecida como NOP sled. Essas

instruções NOP não realizam nenhuma operação quando executadas, servindo como um "escorregador" para o fluxo de execução do programa. Isso significa que, se o endereço de retorno apontar para qualquer posição dentro do NOP sled, o programa seguirá executando até encontrar o shellcode.
Ao executar o programa dentro do ambiente do depurador GDB, o invasor pode verificar o endereço de retorno e confirmar que aponta para o local desejado dentro do NOP sled. Em seguida, o programa é continuado e o shellcode é executado, permitindo ao invasor realizar ações maliciosas.

c) The example runs within the debugger but not outside it, due to the security mechanisms implemented by the runtime. Explain those mechanisms, including address-space layout randomization, stack protector (canary).

R:

- **Address-Space Layout Randomization (ASLR):**

O Address-Space Layout Randomization (ASLR) é uma técnica de segurança que randomiza os endereços de memória usados por um programa, dificultando a previsibilidade desses endereços. Ele embaralha os endereços base dos arquivos executáveis, bibliotecas compartilhadas, heap e pilha, entre outros componentes, cada vez que o programa é executado.
Com o ASLR ativado, os endereços usados pelos componentes do programa, incluindo a pilha e a localização de funções importantes, são randomizados. Isso torna difícil para um invasor determinar os endereços exatos de memória que precisam ser explorados durante um ataque, pois os endereços são diferentes em diferentes instâncias do programa.
No entanto, quando o exemplo é executado dentro do depurador, o ASLR é geralmente desativado para auxiliar na depuração. Isso permite ao invasor prever de forma confiável os endereços de memória e explorar o código vulnerável com sucesso. Mas fora do depurador, o ASLR estaria ativado, tornando significativamente mais desafiador para um invasor explorar a vulnerabilidade.

- **Stack Protector (Canary):**

O Stack Protector é um mecanismo de segurança implementado pelo compilador para detectar estouros de buffer baseados na pilha. Ele coloca um valor aleatório, chamado de Canary, antes do endereço de retorno na pilha. Esse Canary é verificado antes que a função retorne. Se o seu valor for modificado, indicando um estouro de buffer, um erro ou exceção é acionado, impedindo a execução de código malicioso.
Quando o exemplo é executado dentro do depurador, o Canary pode ser desativado ou modificado para facilitar a depuração e o desenvolvimento de exploits. No entanto, fora do depurador, a proteção do Canary estaria ativa, tornando mais difícil para um invasor sobrescrever o endereço de retorno sem acionar uma exceção.

E) Extra question

Considering everything you have learned during the course, evaluate how much you have learned about System Software and how useful you think this is for your knowledge in computer systems.

R: O curso foi um ótimo primeiro contato com os conceitos de software de sistema, cobrindo diversos tópicos pouco explorados nas outras matérias da graduação. Apesar de que por conta da densidade do conteúdo tenha sido difícil ter um aproveitamento total do que foi ensinado durante as aulas, acredito que com mais um pouco de estudo por conta própria e de tentativas e erros com outros exercícios já seja possível se aprofundar mais no tópico.