Fundamentos de Segurança Informática 2021/2021 — Teste Intercalar — 11/12/2021

Duração: 1h:30m

Número mecanográfico: | | | | | | | | | |

Nome completo: _____

Grupo 1 - Conceitos e Terminologia de Segurança Informática *(20%)*

1.1. Indique duas motivações discutidas em aula para um atacante comprometer servidores/*back-ends* que fornecem serviços a um grande número de utilizadores comuns.

1.

2.

1.2. Descreva as duas dimensões de uma matriz de análise de risco como aquela apresentada na figura (a ordem é irrelevante, desde que o conteúdo das células faça sentido).

Horizontal:

Vertical:

1.3. Atribua um número ①: Ameaça, ②: Vulnerabilidade ou ③: Exploit, a cada um dos seguintes exemplos.

	ativista que discorda da política de uma empresa
	mensagem que provoca leitura de memória *out of bounds* na heap
	empresa concorrente
	overflow de inteiros no cálculo de tamanho de buffer
	catástrofe natural
	utilização de memória depois de libertada
	JavaScript que faz heap spraying
	esquecimento de realizar update de segurança

1.4. Explique o conceito de *zero-day vulnerability* e indique uma razão pela qual poderá ter grande valor de mercado.

Grupo 2 - Controlo *(40%)*

2.1. No desenvolvimento de *shell code*, o objetivo é criar uma sequência de bytes que possa ser injetada numa máquina comprometida, e.g., por um *buffer overflow*.

Explique se (e como) as seguintes propriedades de *shell code* facilitam ou dificultam a tomada de controlo:

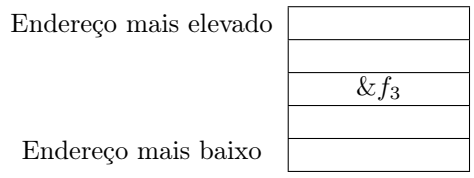
a) O *shell code* contém o valor `0x90`, o código da instrução NOP.

b) O *shell code* contém o byte com valor nulo `0x00`.

2.2. Explique porque é que um erro de gestão de memória *use after free* pode expôr uma vulnerabilidade que permite a um atacante concretizar uma tomada de controlo.

2.3. Recorde o que estudou sobre Return Oriented Programming.

a) Suponha que pretende utilizar Return Oriented Programming para executar as instruções das funções f_1, f_2, f_3, f_4, f_5, por esta ordem. Ilustre como colocaria os endereços dessas funções na stack utilizando o diagrama em baixo, assumindo que não recebem parâmetros. Justifique na caixa ao lado do diagrama.



b) Das 5 funções acima, a qual seria mais fácil passar parâmetros? Explique porquê e indique de que forma poderia passar esses parâmetros à função.

2.4. Explique o que é *Just-in-Time (JIT) Compilation*, e porque é que este tipo de processo pode facilitar ataques de tomada de controlo.

2.5. Considere o seguinte código, que foi a razão para uma vulnerabilidade com elevado nível de gravidade no openSSH.

```
nresp = packet_get_int();
if (nresp > 0) {
   response = xmalloc(nresp*sizeof(char*));
   for (i = 0; i < nresp; i++)
       response[i] = packet_get_string(NULL);
}
```

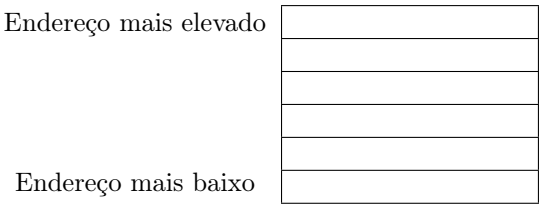
Explique de que forma um atacante controlando o valor da variável nresp pode provocar uma falha na gestão de memória que está na origem do problema.

2.6. Indique, justificando, duas formas de contornar mecanismos de Address Space Layout Randomization.

1.

2.

2.7. Recorde o que estudou sobre mecanismos de proteção com base em canários, e indique no diagrama em baixo como um compilador pode dispor na stack as seguintes zonas, para dificultar ao máximo um ataque por tomada de controlo: ① **variável local: buffer**, ② **variável local: apontador para função**, ③ **canário**, ④ **frame/return address**, ⑤ **parâmetros recebidos**, ⑥ **cópia de parâmetros recebidos**. Justifique a sua resposta na caixa ao lado do diagrama.



2.8. O que é uma *Shadow Stack* e que tipo de ataques permite mitigar? Justifique a sua resposta.

Grupo 3 - Princípios da Segurança de Sistemas e Controlo de Acessos *(10%)*

3.1. Recorde os princípios de segurança de sistemas que estudámos nas aulas de FSI: *economia nos mecanismos*, *proteção por omissão*, *desenho aberto*, *defesa em profundidade*, *privilégio mínimo*, *separação de privilégios*, e *mediação completa*.

a) Enuncie o princípio do *desenho aberto* e apresente uma motivação para a sua utilização.

b) Explique o princípio da *proteção por omissão* e dê um exemplo da sua aplicação correta.

3.2. Apresente um aspeto positivo e um aspeto negativo dos mecanismos de controlo de acessos baseados em *Listas de controlo de acessos*.

Positivo

Negativo

Grupo 4 - Segurança em Sistemas Operativos *(20%)*

4.1. Explique os dois mecanismos fundamentais através dos quais um sistema operativo do tipo Linux implementa o isolamento entre processos e que são formas de ① *system call interposition* e ② *software fault isolation*

1.

2.

4.2. A configuração usual de controlo de acessos a memória limita as permissões concedidas ao próprio Kernel. Dê um exemplo de uma destas limitações e explique porque é imposta, referindo o princípio da segurança de sistemas relevante.

4.3. Recorde o que estudou sobre o controlo de acessos no sistema de ficheiros usual em sistemas Linux.

a) Explique porque é uma forma de *discretionary access control*, e o que o distingue de *mandatory access control*.

b) Apresente uma situação em que um processo altera o seu próprio *effective user id* em tempo de execução de acordo com o princípio do *privilégio mínimo*.

4.4. A utilização de máquinas virtuais levanta a questão sobre se o código a correr em ambientes virtualizados consegue detetar essa forma de confinamento. Apresente dois exemplos práticos em que ① é importante que isto seja dificultado pelo ambiente de virtualização ou ② isto seja facilitado pelo ambiente de virtualização.

1.

2.

Grupo 5 - Malware e Deteção *(10%)*

5.1. Explique a principal diferença entre um *virus* e um *worm*.

5.2. Recorde o que estudou sobre o funcionamento dos anti-vírus actuais.

a) Explique o conceito de deteção baseada em assinaturas (*signature-based*).

b) Identifique uma estratégia que o malware actual utiliza para tentar ultrapassar os mecanismos de deteção baseados em assinaturas.