Sécurité Logicielle

– Examen (1) –

1 Assembleur IA-32 (5 points)

Reconstituez le code C qui correspond au code assembleur suivant :

```
.LC0:
        .string  "i vaut: %d\n"
        .text
main:
        pushl    %ebp
        movl     %esp, %ebp
        subl     $8, %esp
        movl     16(%esp), %ebx
        movl     $.LC0, (%esp)
        movl     %ebx, 4(%esp)
        call     printf
        xorl     %eax, %eax
        cmpl     $0, %ebx
        movl     $-10, %edx
        jle      .L2
.L1:
        imull    %ebx, %eax
        addl     $1, %eax
        cmpl     %eax, %ebx
        jg       .L1
        imull    %ebx, %eax
        addl     %eax, %edx
.L2:
        movl     %edx, %eax
        addl     $8, %esp
        movl     %ebp, %esp
        popl     %ebp
        ret
```

2 Mémoire non exécutable : vers le Return-Oriented-Programming (15 points)

Lisez l'article "*Mémoire non exécutable : vers le Return-Oriented-Programming (ROP)*" par Éric Bourry publié dans Misc #51 (sept/oct 2010). Puis rédigez des réponses aux questions suivantes.

Questions

1. Rappelez rapidement le principe d'un ret-into-libc.
2. Le ret-into-libc permet de contourner certaines des protections listées ensuite, lesquelles et pourquoi : NX/W^R, ASLR, Canaries.

3. À partir de ce qui est écrit dans la section 1.2 (*'Enchaîner plusieurs appels'*) de l'article, dessiner une pile modifiée qui enchaîne un appel à `'system("ls")` puis un appel à `'exit(0)'`. Pour cela, on suppose que l'on dispose des adresses suivantes : `@system`, `@"ls"`, `@exit`.
4. Toujours dans la section 1.2 de l'article, expliquez en détaillant pas à pas une exécution le système du `"leave-ret"` (aussi appelé fonction *'trampoline'*).
5. Dans la section 1.3 (*'Ajouter des octets nuls'*), l'auteur explique comment rajouter des octets nuls. Expliquez en quoi un octet nul est-il problématique et quelle est la méthode suggérée par l'auteur.
6. Dans la section 2.1 (*'Création d'un code avec octets nuls'*), il est question de la PLT. Rappelez brièvement à quoi elle sert et comment elle fonctionne.
7. Par la suite, l'auteur cherche des fonctions dans la libc car elles ne sont pas présentes dans la PLT. Ceci est possible seulement parce qu'il a fait une hypothèse au début de l'article. Quelle est cette hypothèse ? Expliquez.
8. À la section 3.1 (*'Utiliser le code des autres'*) l'auteur parle des architectures IA64. Quelle grosse différence existe-t-il entre IA32 et IA64 concernant le passage des arguments des fonctions ? Et comment l'auteur propose-t-il de contourner cette difficulté ?
9. À la section 3.2 (*'Le Return Oriented Programming (ROP)'*), l'auteur parle d'une nouvelle technique appelée ROP. Expliquez le fonctionnement de cette technique. Dites aussi quelle particularité des instructions x86 rend l'exploitation via des ROPs plus aisée ? Expliquez et justifiez.
10. À la section 3.3 (*'Exemple simple'*), l'auteur expose un exemple. En reprenant la figure 3, détaillez pas à pas le fonctionnement de l'exploit.
11. Enfin, dans la section 3.4 (*'Contournement de l'ASLR'*), l'auteur explique comment déjouer l'ASLR sans avoir recours au brute-force. Expliquez rapidement les points principaux des techniques de brute-force (qui ne sont pas évoquées dans l'article) ainsi que la technique qui est décrite par l'auteur.