Université Joseph Fourier
GRENOBLE
SCIENCES, TECHNOLOGIES, SANTE

Analyse de vulnérabilité et exploitation

WESSI14

Karim Hossen & **Josselin Feist**

Grenoble INP ENSIMAG

Verimag

Plan

- Université de Grenoble
- Analyse de vulnérabilité et exploitation
  - TP exploitation
  - TP post exploitation
  - Demo
- SecurIMAG

Qui ?

- Karim Hossen
  - Doctorant au LIG en fin de thèse
  - Encadré par Roland Groz / Catherine Oriat
  - 3 années d'enseignement (sécurité, cryptographie, programmation python, applications Web)

- Josselin Feist : doctorant à Verimag, 1er année
  - Encadré par Marie Laure Potet / Laurent Mounier
  - 1ère année d'enseignement (sécurité, réseaux)

Université de Grenoble : contexte

- 6 établissements
- 60000 étudiants

- 2 Masters spécialisés en sécurité

Université de Grenoble : Master SAFE/SCCI

- Présentation de Cours/TPs niveau M2
- UJF : Master **SAFE** / Sécurité, Audit, informatique légale, alternance
- UJF / Ensimag : Master **SCCI** / Security , Cryptology and Coding of Information systems (master anglophone)
- Ensimag : **ISI** (3ème année) (seul le premier TP y est donné)

Université de Grenoble : Master SAFE/SCCI

- Environ 15 étudiants pour SAFE / SCCI, 60 pour ISI
- Un cours avec même contenu dans les filières : « Audit et Sécurité »

Analyse de vulnérabilité et exploitation

- 2 TPs de 3H

- But :
  - Pratique concrète de notions souvent théoriques
  - Recherche et déclenchement d'un buffer overflow, exploitation, et création d'un module metasploit et utilisation des payloads plus complexes
  - Utilisation d'outils couramment utilisés dans le domaine :
    Immunity Debugger, Python, Metasploit
  -

1er TP

- Première séance :
- Les étudiants ont un livret de 20 pages à lire avant la première séance.
  - Rappel de notions d'assembleur
    - Cours d'introduction au reverse-engineering si possible
  - Fonctionnement de la pile
  - Etude des buffers overflow
  - Exploitation

- -> Tout le monde à une base minimale

1er TP : Immunity Debugger

- Découverte de Immunity Debugger

- Etude d'un petit programme
  - Pile, tas, instructions, registres
  - Utilisation de breakpoints
  - Modification de la mémoire
  - Comprendre en détails les appels de fonctions

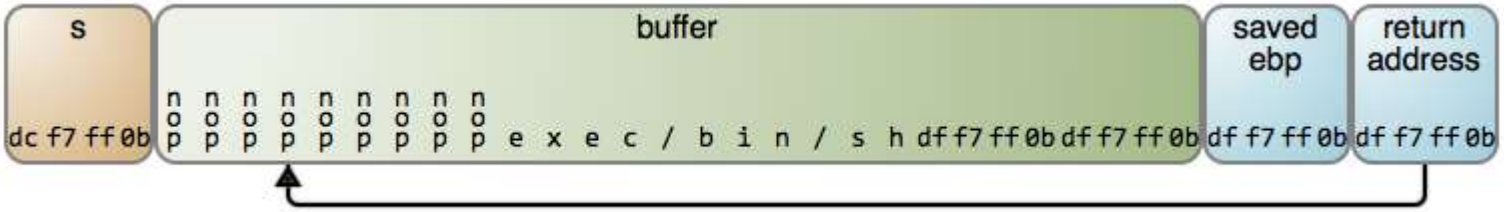
  - Ex: Modifier l'affichage de la fonction *printf* de plusieurs façons différentes

1er TP : Buffer overflow

- Programme plus grand (petit serveur créé pour l'occasion, petit lecteur mp3 avec CVE simple…)
- Script python pour jouer sur les entrées
- Visualisation du buffer overflow depuis le debugger
- Technique de « pattern » avec un script python pour réussir à trouver quels octets de l'entrée font planter le programme

1er TP : Exploitation

- Détournement du flow vers un shellcode
  - Permet au étudiant de bien comprendre la notion « exploiter une vulnérabilité »



  - Qu’est-ce que shellcode ?
  - Ou placer le shellcode ?
  - Comment l’exécuter ?

1er TP : Contournement de protections

- Contournement partiel de la protection ASLR (= adresses rendues aléatoires)

- Contournement de la stack protection (la pile est non exécutable), grâce au ROP *(pour les plus rapides)*

1ER : récapitulatif

- Idée du premier TP :
  - Découverte d'un débuggeur windows
  - Exploitation d'un bof dans des conditions réels (on ne sait pas forcement ce que fait le programme...)
  - Notions de mécanismes de protection / contournement

- Second TP : même programme, même vulnérabilité, mais utilisation de Metasploit

TP2 : Metasploit

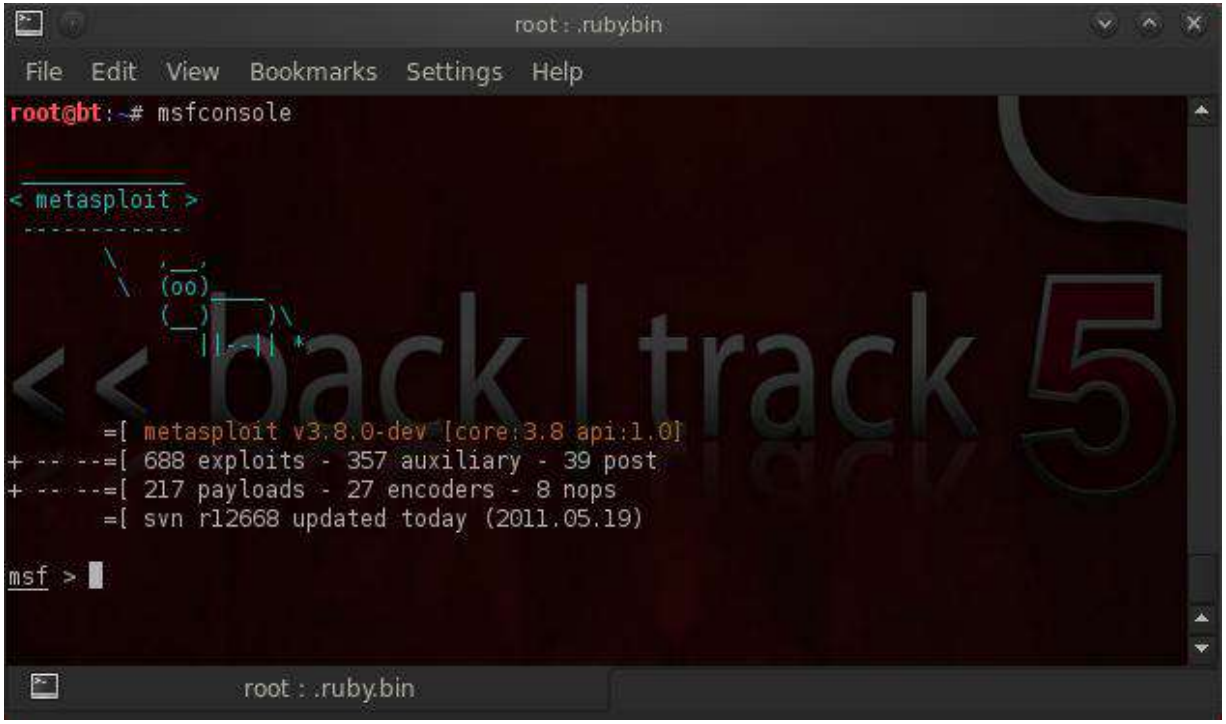
- Le premier TP montre comment exécuter un code arbitraire
- Le second TP s'intéresse à Metasploit
  - Framework
  - Shellcode
    - Comparaison entre un simple Shell et meterpreter
  - Post-exploitation
    - Effacer ces traces
    - Garder un accès sur la machine
    - Explorer le réseau local de la machine
    - …

TP2 : Metasploit

- Organisation
  - Documentation
    - Users guide
    - Developers guide
  - Modules
    - Auxiliary
    - Encoders
    - Exploits
    - Nops
    - Payloads
    - Post
  - Scripts
    - Event_manager, enum_firefox …

TP2 : Utilisation de Metasploit

- Msfconsole



- Msfpayload, Msfencode, MsfCli

TP2 : Création de module

- Création de module à partir d'un exemple

```
class Metasploit3 < Msf::Exploit::Remote

      include Msf::Exploit::Remote::TCP

      def initialize
           super(
                'Name'           => 'Simplified Exploit Module',
                'Description'    => 'This module sends a payload',
                'Author'         => 'My Name Here',
                'Payload'        => {'Space' => 1024, 'BadChars' => "\x00"},
                'Targets'        => [ ['Automatic', {} ] ],
                'Platform'       => 'win',
           )
           register_options( [
               Opt::RPORT(12345)
           ], self.class)
      end

      # Connect to port, send the payload, handle it, disconnect
      def exploit
           connect()
           sock.put(payload.encoded)
           handler()
           disconnect()
      end
end
```

TP2 : Création de module

- A l'aide des fonctions de Metasploit
  - TCP, génération de nops, ..

- Module stable et réutilisable
  - Vérification de l'espace disponible
  - Suppression de certains caractères
  - Utilisation des « targets »
  - Commentaires

TP2 : Utilisation simple

- Payload simple (reverse tcp)
  - Que peut-on faire depuis un Shell Windows ?
    - Créer un compte ?
    - Télécharger un fichier ?
    - Installer un programme ?
    - Prendre une capture d'écran ?
    - Tester la présence d'antivirus ?
  - Le tout doit être fait avec le Shell uniquement
  - But : montrer la limite de ce genre de shellcode

TP 2 : Utilisation avancée

- Meterpreter
  - Shell abstrait (Linux / Win)
  - Commandes puissantes
    - Escalade de privilèges
    - Capture d'écran
    - Keylogger
    - Backdoor
    - Suppression des logs
    - …
  - Facile à utiliser

TP 2 : Utilisation avancée

- Meterpreter
  - Dans quels ordre exécuter ces commandes ?
    - Vérification de l'activité utilisateur
    - Vérification des processus
      - Quitter les antivirus
    - Escalade de privilèges avec un ensemble d'exploits
    - …
    - Trouver le secret !

TP 2 : Récapitulatif

- Utilisation de Metasploit
  - Les divers outils, script et payload du Framework
- Ecriture/compréhension de modules
- Préparation au TP de test de pénétration
  - Par groupe
  - Attaque / Défense

Récapitulation général

- Notion concrète pour les étudiants
  - Voir l'impact d'une vulnérabilité
- Répartir des deux sessions avec une expérience réelle
  - CVE
- Goût à la sécurité ? (ou au moins une meilleure sensibilité)
- Développer des logiciels plus sécurisés

Difficultés

- Mise en place
  - Machine virtuelle
- Temps
  - 6H / nombre d'étudiants
- Programmation Python/Ruby
- Assembleur
  - Notions de base


Demo

SecurIMAG

Fun
Talk
Challenges
Stages & Emplois
Tests d'applications Web
Capture The Flag

Entrainement au CTF
Streaming live
Etude de vulnérabilités
Etude de malwares

Participation au CTF
Ouvert à tous
Gratuit

Google : securimag

SecurIMAG

- 15 étudiants en moyenne
  - Tous niveaux

- Permet de repérer les personnes intéressées par la sécurité
  - Challenge en début d'année

- Pour les étudiants
  - Culture de la sécurité
  - Approfondir des choses pas forcement vues en cours
  - Contacts pour des stages / emplois

Merci pour votre attention