Les frameworks d’audit

Sommaire

- Généralités
- Nos besoins
- Quel framework ?
- CANVAS
- Conclusion

ATHEOS - Fiche d'Identité

- **Cabinet de conseil leader sur son marché**

atheos
L'esprit d'Architecte

- Pôle historique IAM
- Pôle Continuité d'Activité
- **Pôle Audit Sécurité**
- Expertise de plus de 6 ans

- **Faits et Chiffres Clefs :**

- Plus de 60 Experts dédiés
- 100% de clients grands comptes
- Fondateur des RIAM 2007 Les Rencontres de l'Identity & Access Management
- Croissance de plus de 35 % par an
- CA 2006 : 5,5 Millions d'€
- Plus de 60 projets

Généralités

Rappels sur les audits techniques

- Audit interne, audit externe (ou test d'intrusion ou pentest)
- Audit *whitebox*, audit *blackbox*
- Audit manuel, audit automatique

Les frameworks d'audit existants 1/2

Core Impact	- Outil commercial - Société Core Security - Site http://www.coresecurity.com/ - Noyau en C++, modules en python
Canvas	- Outil commercial - Société Immunity Inc. - Site: http://www.immunityinc.com/ - Noyau et modules en python
Metasploit	- Outil opensource - Site: http://www.metasploit.com - Noyau et modules en ruby

Les frameworks d'audit existants 2/2

Ce sont des outils offensifs :

- Ils ne sont pas des scanners de vulnérabilités (Nessus, Qualys, etc)
- Ils n'ont pas de notions de faux positifs/faux négatifs
- Ils n'ont pas pour objectif de faire un état des lieux

Ont ils une utilité ?

Réponse :

Quels sont nos besoins ?

Nos besoins

Les exploits

Si le client autorise l'exploitation des vulnérabilités trouvées sur ses serveurs ou stations de travail :

- Pas besoin d'avoir de 0days
- Exploits les plus fiables possible
- Exploits server side, client side et locaux
- Pas besoin de furtivité

Les outils

Objectif: reproduire la toolbox d'un auditeur

- Scanner réseau
- Scanner de comptes et mots de passe
- Outils de *fingerprinting*
- Etc

Un environnement de développement

Des possibilités d'évolution rapide :

- Ajouter ses propres fonctionnalités :
  - ajout d'outils ou d'exploits
- Améliorer l'existant :
  - ajout de version d'OS supportée dans un exploit

⇒ **Un auditeur sans faire de R&D et de veille techno est un simple utilisateur !**

En résumé

Un seul besoin :

UN UNIQUE OUTIL POUR LES AUDITS

Quel framework ?

~~Core Impact~~

- Pas du tout vendu dans le but d'être un framework de développement :
  - Noyau closed source
  - API non documentée
  - ⇒ Difficile d'ajouter ses propres fonctionnalités
- Très cher !!

~~Metasploit~~

- Framework de développement évolué
- Mais :
  - Depuis quelques mois seulement une GUI sous Windows (très important pour les clients !)
  - Très orienté exploits (peu d'outils)
  - Pas ou peu d'exploits locaux
  - Exploits pas souvent fiables
  - Ruby utilisé
  - Architecture de développement contraignante :
    - Temps de chargement du framework > 30s
    - Multitude de fichiers pour ajouter une fonctionnalité (ex: 12 fichiers pour uniquement Meterpretux)

⇒ Uniquement à utiliser pour certains exploits au cas où …

Canvas

- Pourquoi est-ce le plus utile ?
- En quoi est-il utile ?
- Non je n’ai pas d’actions chez Immunity !

Canvas

D’un point de vue utilisateur 1/2

- Fonctionne essentiellement sous Linux ou Windows
- GUI disponible (en GTK) mais peut fonctionner en mode console
- Simple d’utilisation (ne nécessite pas forcément des connaissances en exploitation de failles)
- Possibilité de lancer un exploit seul (mode console) ou le framework (mode GUI)
- Outils/exploits classifiés
- Des références sur la vulnérabilité pour chaque exploit

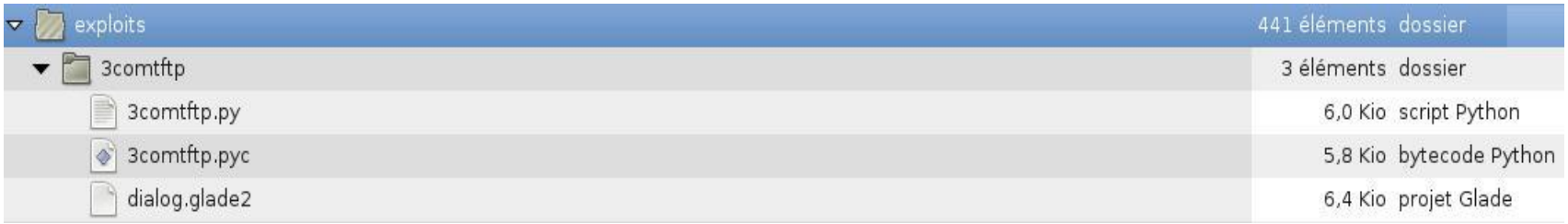
D’un point de vue utilisateur 2/2

- Notion de *target* :
  - Une adresse IP ou une classe réseau : cible pour exécuter un exploit ou un outil
  - Une ou plusieurs *targets* par *Node*

- Notion de *Node* :
  - Notre machine : le *LocalNode*
  - Exploitation réussie sur une machine cible : *LinuxNode* *Win32Node*, etc (MOSDEF)
  - Élévation de privilèges sur les *Nodes*
  - Rebond sur d’autres cibles depuis les *Nodes*
  - Outils post-exploitation

D'un point de vue développeur 1/2

- Entièrement écrit en python
- Exploit = Outil = Module Canvas
- Une unique arborescence pour un module :



- Un répertoire *libs* pour ajouter des librairies Python :
  - Ex. la dernière : Paramiko

- Une documentation très pauvre

D'un point de vue développeur 2/2

- Template d'un module Canvas :

```
#! /usr/bin/env python
import XXX

NAME = XXX
DESCRIPTION = XXX
DOCUMENTATION = XXX
PROPERTY = XXX

class theexploit(canvasexploit):
    def __init__(self):
        canvasexploit.__init__(self)
        […]

   def run(self):
        […]

if __name__=='__main__':
    app = theexploit()
```

Les exploits 1/2

- Des updates à chaque début de mois :
  - un exploit pour la faille OpenSSL sous Debian dans la dernière update
- Beaucoup de plate-formes supportées :
  - Windows (NT, 2000, XP, 2003)
  - Unix (Linux, Solaris, AIX, MacOSX, FreeBSD, Netopia, HP-UX, SCO)
- Fiabilité des exploits :
  - Kostya Kortchinsky et la fiabilité des exploits sous Windows (https://www.blackhat.com/presentations/bh-eu-07/Kortchinsky/Presentation/bh-eu-07-kortchinsky.pdf)
  - Dans le cas contraire :
    - le risque est identique avec les exploits Metasploit ou sur Internet
    - qu'est-ce qui vous empêche de les fiabiliser ?
- Des protections comme DEP sous Windows ou Execshield sous Linux contournées

Les exploits 2/2

- Sociétés éditrices de pack Canvas :
  - GLEG - VulnDisco Pack :
    - http://www.gleg.net/products.shtml
    - très peu d'updates et principalement des 0days
    - ⇒ inutile pour un audit classique
  - DSquare Security – D2 Exploitation Pack :
    - http://www.d2sec.com
    - Beaucoup de modules (environ 100) dont quelques 0days
    - Beaucoup d'outils (nous allons y revenir)
    - ⇒ **Un très bon complément à Canvas**

Les outils 1/2

- Des outils pour les targets :
  - Scanner (scanner de ports, serveurs SMTP, etc)
  - Outils de reconnaissance (BDD Oracle, DCE Dumper, HTTP fingerprint, etc)
  - Brute-forcer (Password Oracle, SMB, etc)
  - Etc.

- Des outils pour les Nodes
  - Outils post-exploitation (Password hash dumper, Keylogger, etc)
  - Elévation de privilèges
  - Etc.

⇒ **Les outils sont nécessaires pour faciliter l'audit**

Les outils 2/2

- Le pack D2 de DSquare ne se focalise pas uniquement sur les exploits
- Il exploite complètement les possibilités de Canvas :
  - d2sec_lotuscan
  - d2sec_clientinsider utilise tous les exploits client side de Canvas

    (http://www.d2sec.com/d2clientinsider.htm)
  - d2sec_sshmosdef utilise Paramiko
    (http://www.d2sec.com/d2sshmosdef.htm)
  - d2sec_smbmosdef utilise l'API DCE-RPC fournie par Canvas
    (http://www.d2sec.com/d2smbmosdef.htm)
  - etc

⇒ **Ces outils sont un très bon exemple de la manière d'utiliser Canvas**

Conclusion

Pourquoi un framework est utile ?

- La notion de tout en un :
  - Des outils
  - Des exploits
  - **Mais un seul outil**

- Un framework sans outils n'est pas utile

- Facilité d'apporter ses propres fonctionnalités :
  - Pourquoi ne pas utiliser scapy dans Canvas par exemple ?
  - Pourquoi ne pas développer un exploit pour une vulnérabilité trouvée lors d'un audit et le réutiliser pour d'autres audits ?
  - Pourquoi ne pas ajouter une version pour un exploit ayant fonctionner lors d'un audit ?

⇒ **Un framework d'audit permet de capitaliser l'expérience acquise au cours des audits**

Contact:

Samuel Dralet - sdralet@atheos.fr