

Sujet d'épreuves des Finales Nationales de la 47^e Compétition des Métiers

MÉTIER N°54 CYBERSECURITE

Module C – Forensic, reverse et revue de code

Soumis par :

Samy SCANNA, Expert WorldSkills France

TABLE DES MATIERES

1.	MODULE C – FORENSIC, REVERSE, REVUE DE CODE.....	3
2.	PLANNING JOURNALIER.....	3
3.	DONNEES TECHNIQUES PARTIE FORENSIC	4
4.	DONNEES TECHNIQUES PARTIE REVERSE.....	5
5.	DONNEES TECHNIQUES PARTIE REVUE DE CODE	6
6.	BARÈME DE CORRECTION	15

1. MODULE C – FORENSIC, REVERSE, REVUE DE CODE

DURÉE TOTALE DE L'ÉPREUVE :

6 heures

L'objectif de ce module est d'évaluer les capacités à enquêter dans des dumps de mémoire et de disque à la recherche d'indices de compromission d'un ou plusieurs serveurs. En complément, vous devrez utiliser des techniques de reverse engineering sur trois « crackme », afin de découvrir les flags qu'ils renferment. Enfin, des portions de code seront fournies pour lesquelles il faudra analyser et corriger les vulnérabilités.

L'accès à internet n'est pas autorisé durant toute la durée du module.

Outils à votre disposition :

VM SIFT Workstation Windows (Forensic) - (login : **sansforensics** / password : **forensics**)

VM Tsurugi-LAB Linux (Forensic) – (login : **tsurugi** / password : **tsurugi**)

VM Kali Linux Full (login : **kali** / password : **kali**)

VM Flare-VM (Reverse) - (login : **flare** / password : **flare**)

2. PLANNING JOURNALIER

	DÉBUT	FIN	TÂCHES	TOTAL
C 2	7h30	8h00	Arrivée des candidats	0h30
	8h00	9h00	Consignes du jury, étude du sujet, et prise en main espace métier	1h00
	9h00	12h00	Épreuve Module C + pause	3h00
	12h00	13h00	Service du déjeuner	1h
	13h00	16h00	Épreuve Module C + pause	3h00
	16h00	18h00	Correction	2h00
TOTAL ÉPREUVE (h)				6h00

3. DONNEES TECHNIQUES PARTIE FORENSIC

Forensic CAS 1 (Windows) :

Profile Volatility : VistaSP1x86

Le serveur web Windows d'une entreprise a fait l'objet d'une intrusion par le biais de son site web. Notre équipe est arrivée juste à temps pour capturer une image disque du système en cours d'exécution et de sa mémoire en vue d'une analyse plus approfondie.

Les fichiers sont fournis sur le bureau de votre poste dans le dossier « **Forensic CAS 1** ».

Vous répondrez aux questions suivantes dans un fichier TXT appelé « **forensic-cas-1.txt** » disposé sur le bureau d'une des deux machines compétiteur.

Q1. Quels sont les types d'attaques qui ont été menées sur cette machine ?

Q2. Quels sont les utilisateurs qui ont été créés par l'attaquant et à quels groupes ont-ils été ajoutés ?

Q3. Quels sont les traces (fichiers, outils, informations, etc.) laissés par l'attaquant ? (en supposant que notre équipe soit arrivée à temps et que l'attaquant n'ait pas pu nettoyer et effacer ses traces)

Q4. Quelle est l'adresse IP qui a été utilisée pour mener les attaques ?

Forensic CAS 2 (GNU/Linux) :

Le serveur web GNU/Linux d'une entreprise a fait l'objet d'une intrusion par le biais de son site web. Notre équipe est arrivée juste à temps pour capturer une image disque du système en cours d'exécution en vue d'une analyse plus approfondie.

Les fichiers sont fournis sur le bureau de votre poste dans le dossier « **Forensic CAS 2** ».

Tips -> Commande Linux pour monter l'image et faire le mappage des volumes :

```
sudo ewfmount image.E01 dossier/
```

```
sudo ls -l dossier/ (récupérer le nom du fichier à l'intérieur)
```

```
sudo kpartx -al dossier/fichier
```

```
sudo kpartx -av dossier/fichier
```

Puis lister les volumes disponibles avec : `sudo lvdisplay`

A vous de jouer.

Vous répondrez aux demandes suivantes dans un fichier TXT appelé « **forensic-cas-2.txt** » disposé sur le bureau d'une des deux machines compétiteur.

Réalisez la liste des événements liés à cette attaque, dans l'ordre de leur dérouler (l'horodatage n'est pas demandé), voici un exemple :

DESCRIPTION :

- Upload d'un webshell via exploitation d'une vulnérabilité de type « file upload ».
- Création de l'utilisateur « toto » et ajout de cet utilisateur au groupe « tata ».

Enfin, complétez en fournissant les numéros des deux CVEs des exploits connus utilisés sur cette machine et l'adresse IP de l'attaquant.

4. DONNEES TECHNIQUES PARTIE REVERSE

Il vous est fourni dans le dossier « Reverse » trois cas appelés « cas1 », « cas2 » et « cas3 ». L'objectif étant simple, trouver le flag pour chacun des cas.

Renseignez l'ensemble des flags trouvés dans un fichier txt disposé sur le bureau d'une des deux machines compétiteur et appelé « **reverse.txt** ».

Vous pourrez utiliser plusieurs des outils suivants: x64dbg, ollyDbg, Edb, Gdb, radare2, ghydra.

5. DONNEES TECHNIQUES PARTIE REVUE DE CODE

Revue de code – CAS 1 (Python) :

Voici le premier code à analyser :

```
1  #!/usr/bin/env python
2
3  # start with setuid-wrapper as serveradmin
4  # sudo setup for serveradmin
5  # serveradmin ALL=(ALL) NOPASSWD:ALL
6
7  import tkinter as tk
8  from tkinter import ttk
9  import os
10 import re
11
12 TITLE = 'Administration Serveur'
13
14 def sanitizeCommandEntry():
15     c1 = commandEntryVar.get()
16     c2 = c1.replace('sudo', '')
17     if (c1 != c2):
18         c2 = c2.replace('sudo', '')
19     c2 = re.sub(r';|\\|\\|\\n', '', c2)
20     commandEntryVar.set(c2)
21
22 def execute():
23     commandEntry.configure(state='disabled')
24     if (commandSelectedVar.get() == ''):
25         sanitizeCommandEntry()
26     command = commandSelectedVar.get() + ' ' + commandEntryVar.get() + ' 2>&1'
27     command = command.strip()
28     textOutput.configure(state='normal')
29     textOutput.delete(1.0, tk.END)
30     textOutput.insert(tk.END, command + '\\n\\n')
31     root.update()
32     pipe = os.popen(command)
33     for line in pipe:
34         textOutput.insert(tk.END, line)
35         root.update()
36     textOutput.configure(state='disabled')
37     commandEntry.configure(state='normal')
38
39 def clear_command_entry():
40     commandEntry.delete(0, tk.END)
```

```

42 if __name__ == "__main__":
43     root = tk.Tk()
44     root.configure(background='silver', padx=20, pady=20)
45     root.title(TITLE)
46     ttk.Style(root).theme_use('classic')
47     ttk.Label(root, text=TITLE, font=('Sans Serif', 20), anchor="center").pack(fill=tk.BOTH, pady=20)
48     commandSelectedVar = tk.StringVar(root, "")
49     commandEntryVar = tk.StringVar(root, "")
50     commandsServer = {
51         'sudo usermod -L -e 1': "Utilisateur : Bloquer [utilisateur]",
52         'sudo usermod -U -e 1': "Utilisateur : Débloquer [utilisateur]",
53         'sudo shutdown': 'Serveur : Arrêter',
54         'sudo poweroff': "Serveur : Arrêter d'URGENCE",
55         'sudo reboot': "Serveur : Redémarrer d'URGENCE",
56         '' : '[Commande Libre]'
57     }
58     for command in commandsServer:
59         radiobutton = ttk.Radiobutton(
60             root,
61             text=commandsServer[command],
62             value=command,
63             variable=commandSelectedVar,
64             command=clear_command_entry
65         )
66         radiobutton.pack(fill=tk.X)
67     commandEntry = ttk.Entry(root, textvariable=commandEntryVar, justify=tk.CENTER)
68     commandEntry.pack(fill=tk.X, pady=20)
69     buttonExecute = ttk.Button(root, text='Exécuter', command = execute)
70     buttonExecute.pack(fill=tk.X)
71     textOutput = tk.Text(root, height=10, width=100, state=tk.DISABLED, bg='silver', fg='black')
72     textOutput.pack(fill=tk.X)
73     buttonQuit = ttk.Button(root, text='Quitter', command=root.quit)
74     buttonQuit.pack(fill=tk.X, pady=20)
75     tk.mainloop()
--

```

Cette application dispose d'une ou plusieurs vulnérabilités :

- Identifiez la/les vulnérabilités et donnez les numéros de ligne concernées.
- Donnez un exemple d'exploitation pour chaque vulnérabilité identifiée.
- Proposez une solution permettant de corriger la/les vulnérabilités en spécifiant les lignes à remplacer /corriger.

Vous indiquerez vos réponses dans un fichier txt disposé sur le bureau d'une des deux machines compétiteur et appelé « **code-review-cas-1.txt** ».

Revue de code – CAS 2 (PHP) – Cas Fonderie :

Contexte

Cette application est utilisée dans la partie industrielle du système d'information d'une fonderie. L'usine produisant régulièrement des pièces pour l'industrie d'armement, cette partie du système est physiquement séparée de l'ERP et les utilisateurs n'ont accès qu'aux informations qui leurs sont strictement nécessaires.

La partie de l'application à étudier est utilisée par les opérateurs pour saisir la ou les pièces à transférer vers la zone d'expédition :

- chaque opérateur est affecté à un et un seul entrepôt
- les opérateurs saisissent les transferts à partir d'un bon d'expédition papier
- les pièces sont numérotées de 10000 à 99999

L'application actuelle s'appuie sur une base de données existante avec comme langue de nommage le français, alors que celle de l'application est l'anglais.

Expédition de Pièces

L'opérateur s'est auparavant authentifié et le système lui a affecté son entrepôt.

Cas nominal

1. L'opérateur saisit la liste des pièces à expédier. Cette liste est constituée des numéros de séries séparés par un virgule (pas d'espace autorisé) ou le numéro de série de la pièce à expédier.
2. L'opérateur valide l'envoi de sa demande.
3. Le système vérifie la validité de la demande et demande confirmation à l'opérateur.
4. L'opérateur confirme sa demande.
5. Le système enregistre la demande et en informe l'opérateur.

Autres cas

Liste malformée

Si la liste saisie ne correspond pas à une liste de numéros de séries respectant la numérotation, le système retourne la liste en alertant à l'aide d'une couleur rouge.

Numéro de série non valide

Si la liste contient un ou plusieurs numéros de série non valides (c'est-à-dire n'existant pas ou n'étant pas en attente d'expédition), le système retourne la liste des numéros de série saisis et indique :

- le nombre de numéros de série non valides
- la liste des numéros de série valides sous forme d'un tableau

Structure des fichiers de l'application :

```
|— app.css
|— app.sqlite
|— common.php
|— shipping-post-00-00.php
|— shipping-post-10-00.php
|— shipping-post-10-10.php
|— shipping-post-20-00.php
|— web-root
|   |— shipping.php
```

Structure de la base de données :

```
sqlite> .schema
CREATE TABLE piece
(
    numero_serie INTEGER PRIMARY KEY,
    entrepot TEXT,
    transfert_zone_expedition INTEGER default 0,
    -- 0: en attente, 1: demandé, 2: effectué
    check ( transfert_zone_expedition in(0, 1, 2) ),
    -- A to E: entrepôt
    check ( entrepot in('A','B','C','D','E') )
);
```

Documentation au format numérique disponible dans le répertoire « **Code Review** » pour PHP et SQLite.

Code app.css :

```
1  html {
2  |     background-color: white;
3  | }
4
5  * {
6  |     font-family: monospace;
7  |     cursor: default;
8  | }
9
10 .error {
11 |     background-color: lightcoral;
12 | }
13
14 .well-formed {
15 |     background-color: lightcyan;
16 | }
17
18 .valid {
19 |     background-color: greenyellow;
20 | }
21
22 *:not(h1, h2, h3) {
23 |     font-size: 1rem;
24 |     padding: 0.25rem;
25 | }
26
27 table {
28 |     border-collapse: collapse;
29 | }
30
31 td,
32 th {
33 |     border: solid 1px;
34 |     text-align: center;
35 |     padding: 0.5rem;
36 | }
37
38 th {
39 |     background-color: lightcyan;
40 | }
```

Code common.php :

```

1  <?php
2
3  function unexpectedApplicationBehavior($data = null) {
4      // TODO log
5      session_destroy();
6      die();
7  }
8
9  // dev
10 ini_set('display_errors', 1);
11 ini_set('display_startup_errors', 1);
12 error_reporting(E_ALL);
13 define('PDO_ERRORMODE', PDO::ERRMODE_EXCEPTION);
14 // prod
15 /*
16 ini_set('display_errors', 0);
17 define('PDO_ERRORMODE', PDO::ERRMODE_SILENT);
18 */
19
20 define('ACTION',$_SERVER['PHP_SELF']);
21
22 session_start();
23
24 // storage area operator session
25 $_SESSION["employee.number"] = 314;
26 $_SESSION["employee.roles"] = ["storage_area_operator"];
27 $_SESSION["employee.storage_area_operator_of"] = "B";
28
29 $pdoPlant = new \PDO(
30     'sqlite:../app.sqlite',
31     null,
32     null,
33     [
34         \PDO::ATTR_EMULATE_PREPARES => false,
35         \PDO::ATTR_ERRMODE => PDO_ERRORMODE,
36         \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC
37     ]
38 );
39 ?>
40 <style>
41 <?php require('app.css') ?>
42 </style>
43
44

```

Code shipping.php :

```

1  <?php
2  require('../common.php');
3
4  if (!in_array('storage_area_operator', $_SESSION['employee.roles'])) {
5      unexpectedApplicationBehavior();
6  }
7
8  switch ($_SERVER['REQUEST_METHOD']) {
9      case "POST":
10         require('../shipping-post-00-00.php');
11         break;
12      case "GET":
13         $lpdPost = "";
14         $lpdRO = "";
15         $lpdClass = "";
16         $submitText = "Valider";
17         $submitClass = "";
18         $dataErrors = "";
19         $confirmation = "";
20         break;
21      default:
22         session_destroy();
23         die();
24  }
25  ?>
26  <h1>Expédition de Pièces</h1>
27  <h2>Entrepôt : <?= $_SESSION["employee.storage_area_operator_of"] ?></h2>
28  <h2>Opérateur : <?= $_SESSION["employee.number"] ?></h2>
29  <?= $confirmation ?>
30  <form method="post">
31  <h3>Liste des Pièces à Expédier</h3>
32      <input type="text" id="lpd" size="65" value="<?= $lpdPost ?>" name="lpd" <?= $lpdRO ?>
33      | class="<?= $lpdClass ?>" >
34      <br><br>
35      <input type="submit" id="submit" name="state" value="<?= $submitText ?>" class="<?= $submitClass ?>" >
36      <?= $dataErrors ?>
37  </form>

```

Code shipping-post-00-00.php :

```
1  <?php
2  $postState = $_POST['state'];
3  $lpdPost = $_POST['lpd'];
4
5  switch ($postState) {
6      case "Valider":
7          require('shipping-post-10-00.php');
8          break;
9      case "Confirmer":
10         require('shipping-post-20-00.php');
11         break;
12     default:
13         unexpectedApplicationBehavior();
14 }
```

Code shipping-post-10-00.php :

```
1  <?php
2  $confirmation = "";
3  $sqlShippingCondition = "numero_serie in ($lpdPost) and entrepot = '$_SESSION[\"employee.storage_area_operator_of\"]' and transfert_zone_expedition = 0";
4  if (!preg_match('/^[1-9][0-9]{4}([,][1-9][0-9]{4})*/', $lpdPost)) {
5      // KO submit is not an serial numbers list
6      $lpdClass = "error";
7      $lpdRO = "";
8      $submitText = "Valider";
9      $submitClass = "error";
10     $dataErrors = "";
11 } else {
12     require('shipping-post-10-10.php');
13 }
```

Code shipping-post-10-10.php :

```

1  <?php
2  // okay submit is an serial numbers list
3  $lpdClass = "well-formed";
4  $sql = "
5      select count(*) as count
6      from piece
7      where $sqlShippingCondition
8  ";
9  $numberOfValidPieces= (int) $pdoPlant->query($sql)->fetch()['count'];
10 $numberPiecesFromList = substr_count($lpdPost, ',') + 1;
11 $numberOfInvalidPieces = $numberPiecesFromList - $numberOfValidPieces;
12 if ($numberOfInvalidPieces === 0) {
13     // okay all pieces are valid
14     $lpdRO = "readonly";
15     $lpdClass = "valid";
16     $submitText = "Confirmer";
17     $submitClass = "valid";
18     $dataErrors = "";
19 } else {
20     // KO show number of invalid pieces and the list of valid pieces
21     $dataErrors = "<h4 id='dataErrors'>Nombre de numéros de série invalides : <span class='error'>$numberOfInvalidPieces</span></h4>";
22     $dataErrors .= "<h4>Pièces Valides</h4>";
23     $sql = "
24         select numero_serie
25         from piece
26         where $sqlShippingCondition
27         order by numero_serie
28     ";
29     $validPieces = $pdoPlant->query($sql)->fetchAll();
30     $dataErrors .= "<table><tr><th>Numéros de Série</th></tr>";
31     foreach ($validPieces as $validPiece) {
32         $dataErrors .= "<tr><td>{$validPiece['numero_serie']}</td></tr>";
33     }
34     $dataErrors .= "</table>";
35     $lpdRO = "";
36     $lpdClass = "well-formed";
37     $submitText = "Valider";
38     $submitClass = "error";
39 }

```

Code shipping-post-20-00.php :

```

1  <?php
2  $sql= "
3      update piece
4      set transfert_zone_expedition = 1
5      where numero_serie in ($lpdPost)
6  ";
7
8  $pdoPlant->exec($sql);
9
10 $confirmation = "<h4 id=confirmation>Demande de Transfert enregistrée pour : <span class='valid'>$lpdPost</span></h4>";
11 unset($_SESSION['lpd']);
12
13 $lpdPost = "";
14 $lpdRO = "";
15 $lpdClass = "";
16 $submitText = "Valider";
17 $submitClass = "";
18 $dataErrors = "";

```

Cette application dispose d'une ou plusieurs vulnérabilités :

- Identifiez la/les vulnérabilités et donnez les numéros de ligne concernées et le nom du fichier concerné.
- Donnez un exemple d'exploitation pour chaque vulnérabilité identifiée.
- Proposez une solution permettant de corriger la/les vulnérabilités en spécifiant les lignes à remplacer /corriger.

Vous indiquerez vos réponses dans un fichier txt disposé sur le bureau d'une des deux machines compétiteur et appelé « **code-review-cas-1.txt** ».

6. BARÈME DE CORRECTION

Grille avec le détail des critères de notation objectifs et jugements.

CYBERSECURITE – N°54						
Critère	Sous Critère	Jour	Intitulé du critère de notation	Objectif ou Jugement	Barème	Coef.
A			Module C : Reverse, Forensic, Revue de code		25	
			Sous-titre ou explication du critère			
A	01	2	Forensic CAS 1 : Windows	O	5	1
A	02	2	Forensic 2 : Linux	O	5	1
A	03	2	Reverse : CAS 1	O	1	1
A	04	2	Reverse : CAS 2	O	2	1
A	05	2	Reverse : CAS 3	O	5	1
A	06	2	Revue de code : CAS 1	J	3	1
A	06	2	Revue de code : CAS 2	J	4	1
TOTAL					25	