LISV Laboratoire d'ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Programmation sécurisée (Secure coding)
TP 1
Utilisation d'une chaîne de formatage contrôlée par une source externe

Halim Djerroud

révision 1.0

1 – CWE-134 –

Exercice 1 : Chaînes et Formatage

Écrivez un programme qui demande à l'utilisateur son nom et son âge, puis affiche un message formaté de manière sécurisée.

Exercice 2 : Validation des Entrées

Écrire un programme qui demande à l'utilisateur de saisir un nombre entier entre 1 et 100 inclus. Si l'utilisateur entre autre chose, affichez un message d'erreur et redemandez l'entrée.

Exercice 3 : Calculatrice Sécurisée

Écrire programme calculatrice qui permet à l'utilisateur d'effectuer une addition, une soustraction, une multiplication ou une division. Ne pas utiliser `eval`.

Exercice 4 : Journalisation d'Activités Utilisateur

Écrire un programme qui journalise les actions des utilisateurs dans un fichier `log.txt` avec un horodatage.

Exercice 5 : Analyseur de Fichiers Sécurisé

Écrire un programme qui lit un fichier CSV contenant des produits et calcule leur valeur totale (prix × quantité).

Produit	**Prix**	**Quantité**
Stylo	1.5	100
Cahier	3.2	50
Trousse	8.5	20
Sac	25.0	10
Livre	15.0	5

TABLE 1 – Exemple de contenu du fichier CSV `produits.csv`

Exercice 6

Etudier le code suivant :

```
# SPDX-FileCopyrightText: OpenSSF project contributors
# SPDX-License-Identifier: MIT
""" Non-compliant Code Example """
import sys

# Simulating a global include of sensitive information:
ENCRYPTION_KEY = "FL4G1"

# Simulating a include per language:
MESSAGE = "Contract '{0.instance_name}' created for "


class MicroService:
    """Fancy MicroService"""
    def __init__(self, instance_name):
        self.instance_name = instance_name


def front_end(customer):
    """Display service instance"""
    message_format = MESSAGE + customer
    mc = MicroService("big time microservice")
    print(message_format.format(mc))


######################
# exploiting above code example
######################
if __name__ == "__main__":
    if len(sys.argv) > 1:  # running from command line
        # you can print the global encryption key by using
        #'{0.__init__.__globals__[ENCRYPTION_KEY]}' as
        # argument.
        front_end(sys.argv[1])
    else:
        # running in your IDE, simulating command line:
        # Printing the ENCRYPTION_KEY via the global accessible object
        front_end("{0.__init__.__globals__[ENCRYPTION_KEY]}")
```

2 – CWE-197 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-197

```
value = float(0.0)
while value <= 1:
    print(value)
    value = value + float(1.0/9.0)
```

Exercice 2 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-197

```
value = float(1.0) + float("1e-18")
target = float(1.0) + float("1e-17")
while value <= target:
    print(value)
    value = value + float("1e-18")
```

3 – CWE-754 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-754

```
import sys


class Package:
    def __init__(self):
        self.package_weight = float(1.0)

    def put_in_the_package(self, object_weight):
        value = float(object_weight)
        print(f"Adding an object that weighs {value} units")
        self.package_weight += value

    def get_package_weight(self):
        return self.package_weight


######################
# exploiting above code example
######################
package = Package()
print(f"\nOriginal package's weight is {package.get_package_weight():.2f} units\n")
for item in [sys.float_info.max, "infinity", "-infinity", "nan"]:
    try:
        package.put_in_the_package(item)
        print(f"Current package weight = {package.get_package_weight():.2f}\n")
    except Exception as e:
        print(e)
```

4 – CWE-1109 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-1109

```
numbers = ["one", "two", "three"]

print(f"len({numbers}) == {len(numbers)}")

def len(x):
    """ implementing a dodgy version of a build in method """
    return sum(1 for _ in x) + 1

print(f"len({numbers}) == {len(numbers)}")
```

5 – CWE-1339 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-1339

```
balance = 3.00
item_cost = 0.33
item_count = 5

####################
# exploiting above code example
####################
print(
    f"{str(item_count)} items bought, ${item_cost} each. "
    f"Current account balance: ${str(balance - item_count * item_cost)}"
)
```

6 – CWE-392 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-392

```
import math
from concurrent.futures import ThreadPoolExecutor


def get_sqrt(a):
    return math.sqrt(a)


def run_thread(var):
    with ThreadPoolExecutor() as executor:
        return executor.submit(get_sqrt, var)

####################
# exploiting above code example
####################
# get_sqrt will raise ValueError that will be suppressed by the ThreadPoolExecutor
arg = -1
result = run_thread(arg)  # The outcome of the task is unknown
```

7 – CWE-230 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-230

```
def balance_is_positive(value: str) -> bool:
    """Returns True if there is still enough value for a transaction"""
    _value = float(value)
    if _value == float("NaN") or _value is float("NaN") or _value is None:
        raise ValueError("Expected a float")
    if _value <= 0:
        return False
    else:
```

```
        return True


####################
# attempting to exploit above code example
####################
print(balance_is_positive("0.01"))
print(balance_is_positive("0.001"))
print(balance_is_positive("NaN"))
```

8 – CWE-838 –

Exercice 1 :

Corriger le code suivant pour qu'il soit compatible avec : CWE-838

```
def report_record_attack(stream: bytearray):
    print("important text:", stream.decode("utf-8"))

####################
# attempting to exploit above code example
####################
payload = bytearray("user:  attempted a directory traversal".encode("utf-8"))
# Introducing an error in the encoded text, a byte
payload[3] = 128
report_record_attack(payload)
```