Something Awesome Project for COMP6443: Padding Oracle Attack

Student Information: z5110172 Weitao Wang

Introduction of Project:

A padding oracle is a function of an application which decrypts encrypted data provided by the client, e.g. internal session state stored on the client, and leaks the state of the validity of the padding after decryption. The existence of a padding oracle allows an attacker to decrypt encrypted data and encrypt arbitrary data without knowledge of the key used for these cryptographic operations. This can lead to **leakage of sensible data** or to **privilege escalation vulnerabilities**, if integrity of the encrypted data is assumed by the application.

Block ciphers encrypt data only in blocks of certain sizes. Block sizes used by common ciphers are 8 and 16 bytes. Data where the size doesn't match a multiple of the block size of the used cipher has to be padded in a specific manner so the decryptor is able to **strip the padding**. A commonly used padding scheme is **PKCS#7**. It fills the remaining bytes with the value of the padding length.

In Web Security area, A Web hack that can endanger online banking transaction is ranked the **No. 1 new Web hacking technique for 2010** in a **top 10** list selected by a panel of experts and open voting.Called the **Padding Oracle Crypto Attack**, the hack takes advantage of how Microsoft's Web framework **ASP.NET** protects AES encryption cookies.

Hence I am going to self-learned what is Padding Oracle Attack and how it works in my something awesome project.

Initial Goals:

Using Python to implement how Padding Oracle Attack work.

Studying Methodology:

Youtube video, online blogs, programming, security engineering peers discussion, reading textbook.

Difficulties Analysis:

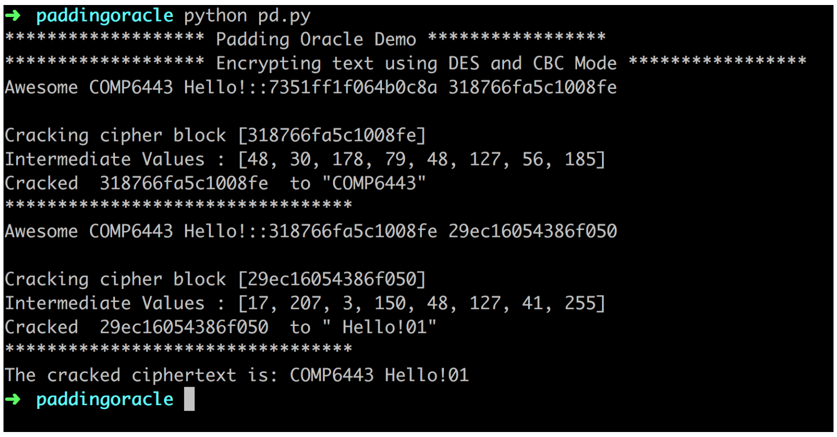
CBC Mode Encryption, PKC#7 Standard, Padding Operation, Oracle Attack, Initialisation Vector, DES, Intermediate Value.

Demonstration:

I used the python 2.7 to implement how padding oracle attacks cracking the cipher text to the plaintext

GitHub link: https://github.com/StephenLover/COMP6443SomethingAwesome

Demo Screen Shot:

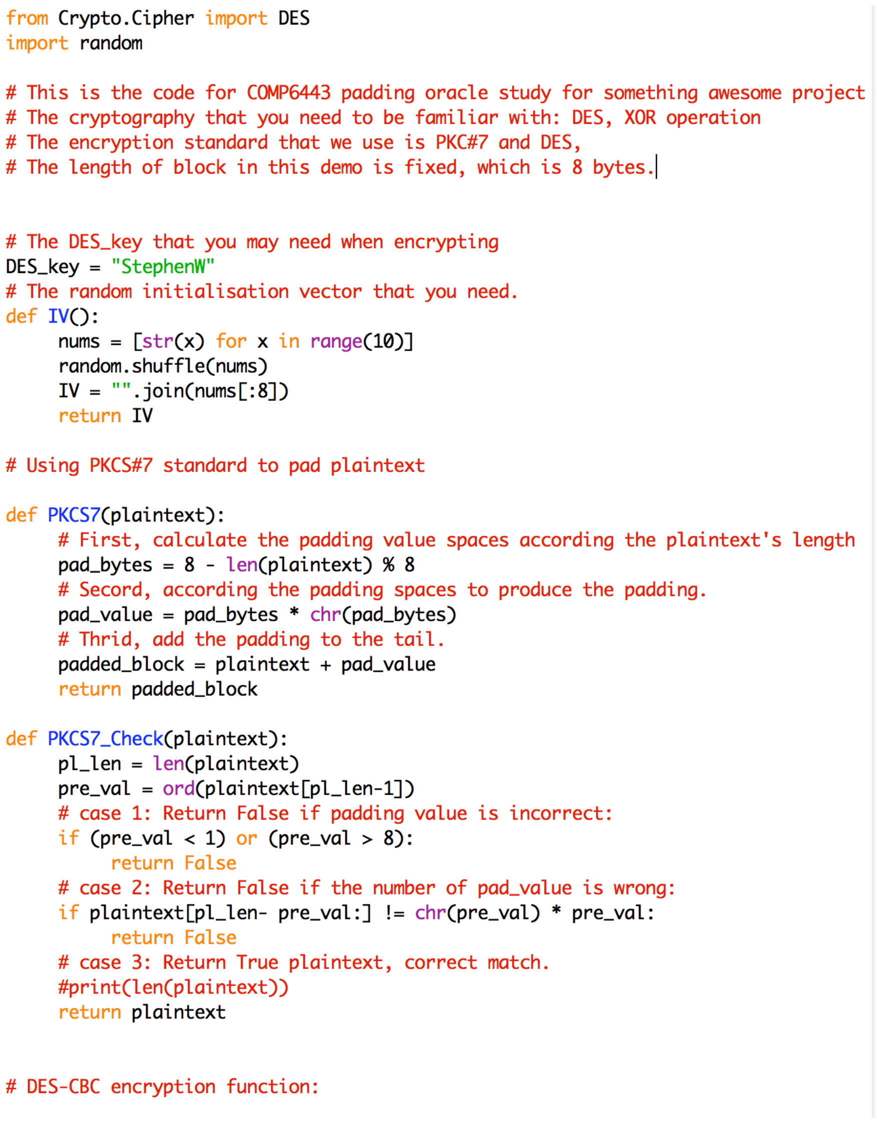


```
➜ paddingoracle python pd.py
****************** Padding Oracle Demo *****************
****************** Encrypting text using DES and CBC Mode *****************
Awesome COMP6443 Hello!::7351ff1f064b0c8a 318766fa5c1008fe

Cracking cipher block [318766fa5c1008fe]
Intermediate Values : [48, 30, 178, 79, 48, 127, 56, 185]
Cracked  318766fa5c1008fe  to "COMP6443"
*********************************
Awesome COMP6443 Hello!::318766fa5c1008fe 29ec16054386f050

Cracking cipher block [29ec16054386f050]
Intermediate Values : [17, 207, 3, 150, 48, 127, 41, 255]
Cracked  29ec16054386f050  to " Hello!01"
*********************************
The cracked ciphertext is: COMP6443 Hello!01
➜ paddingoracle
```

Code Screen Shot:



```python
from Crypto.Cipher import DES
import random

# This is the code for COMP6443 padding oracle study for something awesome project
# The cryptography that you need to be familiar with: DES, XOR operation
# The encryption standard that we use is PKC#7 and DES,
# The length of block in this demo is fixed, which is 8 bytes.


# The DES_key that you may need when encrypting
DES_key = "StephenW"
# The random initialisation vector that you need.
def IV():
     nums = [str(x) for x in range(10)]
     random.shuffle(nums)
     IV = "".join(nums[:8])
     return IV

# Using PKCS#7 standard to pad plaintext

def PKCS7(plaintext):
     # First, calculate the padding value spaces according the plaintext's length
     pad_bytes = 8 - len(plaintext) % 8
     # Secord, according the padding spaces to produce the padding.
     pad_value = pad_bytes * chr(pad_bytes)
     # Thrid, add the padding to the tail.
     padded_block = plaintext + pad_value
     return padded_block

def PKCS7_Check(plaintext):
     pl_len = len(plaintext)
     pre_val = ord(plaintext[pl_len-1])
     # case 1: Return False if padding value is incorrect:
     if (pre_val < 1) or (pre_val > 8):
          return False
     # case 2: Return False if the number of pad_value is wrong:
     if plaintext[pl_len- pre_val:] != chr(pre_val) * pre_val:
          return False
     # case 3: Return True plaintext, correct match.
     #print(len(plaintext))
     return plaintext


# DES-CBC encryption function:
```

Reflection:

In this project, I found my passion about cryptography and web security during padding oracle attack research.

And I realised that the **theoretic knowledge** are extremely important in web security, for example, if you want to know how padding oracle attacks works, you have to be familiar with the CBC Encryption Mode, PKCS7 Standard, DES….

Also, I found the Padding Oracle vulnerability are very common, even exists in the big framework such as ASP.NET.

Reference and Tools:

https://blog.gdssecurity.com/labs/2010/9/14/automated-padding-oracle-attacks-with-padbuster.html

https://github.com/mpgn/Padding-oracle-attack

https://www.acunetix.com/vulnerabilities/web/asp-net-padding-oracle-vulnerability

https://docs.secureauth.com/display/KBA/ASP.NET+Padding+Oracle+Vulnerability

https://www.cio.com.au/article/374310/top_10_web_hacking_techniques_2010_revealed/?fp=4&fpid=2117012706

- Padding Oracle Attack Online Demo: http://erlend.oftedal.no/blog/poet/
- PadBuster: https://github.com/GDSSecurity/PadBuster