МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ

о прохождении учебной практики по получению первичных

профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности

на базе Высшей школы компьютерных наук и прикладной математики образовательно-научного кластера "Институт высоких технологий"

Выполнили: Куртев Артем Юрьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студенты очной формы обучения 3 курса

специальности 10.05.01 Компьютерная безопасность

специализация «Математические методы защиты информации»

Руководитель практики от университета

доцент ОНК «ИВТ» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Киршанова Е.А.

г. Калининград 2023 г.

**СОДЕРЖАНИЕ**

Введение………………………………………………………………………..3

Глава 1. Индивидуальное задание на практику………………………………4

Глава 2. Выполнение задания…………………………………………………5

Глава 3. Подготовка презентации……………………………………………..7

Заключение…………………………………………………………………….8

Список литературы…………………………………………………………….9

Приложения…………………………………………………………………..10

**Введение**

В ведении формулируются цель прохождения практики, а также комплекс задач, которые поставлены перед практикантом руководителем практики на период ее прохождения.

Объем введения 1 – 1,5 страницы.

Вид практики – учебная практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности, далее Учебная практика.

Цель учебной практики: получить первичные профессиональные умения и навыки, в том числе навыки научно-исследовательской деятельности.

Задачи учебной практики:

* Необходимо выбрать и решить одну из задач, представленных в разделе CTF Archive на сайте cryptohack.org
* После решения задачи подготовить и загрузить в свой git-репозиторий презентацию в Beamer, программный код на Python (или другом языке), решающий задачу и данный отчет о выполнении практики.

**ГЛАВА 1. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ**

Нами было выбрано задание под названием Calm Down (HKCERT CTF).

Текст задания выглядит следующим образом:

I am so excited having a chance talking to Alice. She told me to calm down - and sent me an encrypted secret.

Challenge contributed by Mystiz

Connect at nc archive.cryptohack.org 53580

Challenge files:

- chall.py

В файле содержится исходный код задания, позволяющий понять принцип работы системы выдачи задания и проверки введенных данных. (Приложение 1)

Суть задания становится понятной, когда производится подключение указанному адресу. Дан открытый ключ и шифротекст, необходимо расшифровать сообщение, в котором, по всей видимости, содержится необходимый нам набор символов (далее – флаг). По всем внешним признакам, мы имеем дело с RSA.

Обратим внимание, что открытый ключ, а, как следствие и шифротекст, генерируются случайным образом каждый раз, когда устанавливается соединение с сервером.

**ГЛАВА 2. ВЫПОЛНЕНИЕ ЗАДАНИЯ**

В процессе поиска информации о различных атаках на данный алгоритм, а также изучая исходный код, предложенный нам в файле задания, мы обнаружили интересную особенность в реализации алгоритма в наших условиях. При отправке какого-либо сообщения в формате base64, сервер давал ответ, оканчивается ли сообщение точкой или нет. Это подтолкнуло нас на использование атаки, использующей наименьший значащий бит. Иначе говоря, благодаря информации о наличии точки в конце текста можно было дешифровать выданный нам шифротекст.

Из открытого ключа необходимо было достать пару чисел, e и n – степень, в которую возвели секретное сообщение и модуль соответственно. Число n было зашифровано в байтовом виде, переведенном в кодировку base64. Число e удалось достать исходя из спецификации реализации алгоритма на сервере. Использовалось стандартное значение e функции RSA.generate(), равное 65537.

Теперь у нас есть все для того, чтобы иметь возможность повторить зашифрование, ведь , однако расшифровать мы не сможем, так как не имеем d: .

Умножим наш шифротекст на константу . Получим:

Тогда, сервер при расшифровании получит sm вместо оригинального секретного сообщения.

Следующим шагом обратим внимание на то, что сообщение, в нашем случае флаг, состоящий из порядка 20-30 символов, намного меньше модуля n. Иначе говоря, . Установим значение s такое, чтобы sm было чуть больше n. Тогда будет выполнится следующее условие: . Иначе это можно записать так: , и чем ближе к n будет произведение sm, тем меньше будет становиться r, пока не выяснится, что . Из этой формулы уже можно извлечь секретное сообщение: .

Задача сводится к тому, чтобы найти такое s, при котором вышеупомянутая формула будет истинна. В этом нам поможет определение точки в конце текста. Точка в конце кодируется байтом 0x2e. Учитывая, что последний байт n не может быть равен 0x00, так как n – нечетное, а значит необходимо найти такой байт, который при умножении на 0х2е будет выдавать подобное на 0х2е число. Таким байтом является 0х81. Умножив два байта, получим 0x172e. Так, постепенно увеличивая такое число, мы сможем понять, когда мы перешли за границу модуля, ведь точка в конце перестанет быть точкой, а значит сервер сообщит нам об этом. А затем нам достаточно постепенно уменьшить число s, чтобы найти то самое наименьшее s, чтобы декодировать сообщение.

Подводя итог, алгоритм действий следующий:

1. Будем приписывать число 0хf слева к 0х81, пока не получим сообщение о том, что сообщение не имеет точки в конце.
2. После этого, начнем уменьшать число на 1, начиная со старшего разряда, пока вновь не получим сообщение о том, что точка присутствует. В этом случае увеличим тот же бит обратно на 1, чтобы чуть-чуть превышать n.
3. Повторим 2й пункт до тех пор, пока не проверим каждый разряд числа. В итоге будем иметь минимальное s, позволяющее раскрыть секретное сообщение.
4. Расшифруем секретное сообщение, поделив n на s.

Перейдем от теории к практике.

На языке программирования Python напишем программу, способную подключиться к нужному нам серверу и имитировать запросы так, как это могли бы делать мы своими руками. Автоматизируем процесс общения с сервером потому, что в ином случае нам вручную пришлось бы перебирать огромное количество входных данных.

Затем, изложим представленный выше алгоритм в виде кода. Для простоты дальнейшего восприятия дополним его комментариями и поясняющими фразами в окне вывода. Конечный вариант кода представлен в Приложении 2.

По завершении работы программы получаем следующий флаг:

Hey, congratulations on solving the challenge. But please hkcert20{c4lm\_d0wn\_4nd\_s0lv3\_th3\_ch4llen9e}.

Введем набор символов, начинающийся с префикса hkcert20 в необходимое поле на сайте cryptohack.org. Появившаяся звезда в левом верхнем углу задания сигнализирует, что секретное сообщение расшифровано верно, а задание выполнено.

**ГЛАВА 3. ПОДГОТОВКА ПРЕЗЕНТАЦИИ**

По условию задачи практики необходимо было представить презентацию в LaTeX. Для этого мы воспользовались онлайн-редактором Overleaf.

Создадим в нем проект и используем мануал, представленный на смежном сайте.

Результат можно увидеть в git-репозитории по ссылке:

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения данной работы мы познакомились с возможными атаками на RSA, приобрели навыки создания презентаций в LaTeX, закрепили навык создания отчетов по работе.

Все задачи работы успешно выполнены, а цель достигнута.

**СПИСОК ЛИТЕРАТУРЫ**

**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

1. RSA — PyCryptodome 3.18.0 documentation. – Режим доступа: https://pycryptodome.readthedocs.io/en/latest/src/public\_key/rsa.html
2. Объясните процедуру атаки RSA в CTF - Русские Блоги. – Режим доступа: https://russianblogs.com/article/90211187199/
3. RSA least significant bit oracle attack. – Режим доступа: https://crypto.stackexchange.com/questions/11053/rsa-least-significant-bit-oracle-attack
4. Beamer - Overleaf, Online LaTeX Editor. – Режим доступа: https://www.overleaf.com/learn/latex/Beamer
5. Overleaf, Online LaTeX Editor. – Режим доступа: https://ru.overleaf.com/project/
6. GitHub: Let’s build from here · GitHub. – Режим доступа: https://github.com

**ПРИЛОЖЕНИЯ**

**Приложение 1**

import base64

import binascii

import hashlib

import os

import random

from Crypto.PublicKey import RSA

from Crypto.Util.number import bytes\_to\_long, long\_to\_bytes

from secret import message

class RSAKey:

def \_\_init\_\_(self, e, d, n):

self.e = e

self.d = d

self.n = n

def encrypt(self, message):

message = bytes\_to\_long(message)

ciphertext = pow(message, self.e, self.n)

return long\_to\_bytes(ciphertext)

def decrypt(self, ciphertext):

ciphertext = bytes\_to\_long(ciphertext)

message = pow(ciphertext, self.d, self.n)

return long\_to\_bytes(message)

class Challenge:

def \_\_init\_\_(self, message):

self.generate\_key()

self.message = message

def generate\_key(self):

key = RSA.generate(2048)

self.key = RSAKey(key.e, key.d, key.n)

def get\_public\_key(self):

n = base64.b64encode(long\_to\_bytes(self.key.n)).decode()

print(f'[pkey] {n}')

def get\_secret\_message(self):

ciphertext = self.key.encrypt(self.message)

ciphertext = base64.b64encode(ciphertext).decode()

print(f'[shhh] {ciphertext}')

def send(self, ciphertext):

ciphertext = base64.b64decode(ciphertext)

message = self.key.decrypt(ciphertext)

if message[-1:] != b'.':

raise Exception('Be polite. Your message should terminate with a

full-stop.')

print('nice')

def main():

c = Challenge(message)

while True:

command = input('[cmd] ').split(' ')

try:

if command[0] == 'send':

c.send(command[1])

elif command[0] == 'pkey':

c.get\_public\_key()

elif command[0] == 'read':

c.get\_secret\_message()

except:

print('nope')

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Приложение 2**

import base64

import math

from Crypto.Util.number import bytes\_to\_long, long\_to\_bytes

from pwn import \*

#Функция ниже описывает порядок проверки наличия точки в конце сообщения

def orcale\_ifValid(msg\_long):

msg\_base64 = base64.b64encode(long\_to\_bytes(msg\_long))

msg\_full = b'send ' + msg\_base64

proc.sendline(msg\_full)

data = proc.recvuntil(b'[cmd] ')

result = data.replace(b'\n[cmd] ', b'')

return (result == b'nice')

#Считывание публичного ключа с сервера

proc = remote('archive.cryptohack.org', '53580')

data = proc.recvuntil(b'[cmd] ')

proc.sendline(b'pkey')

data = proc.recvuntil(b'[cmd] ')

data = data.replace(b'[pkey] ', b'')

n\_base64 = data.replace(b'\n[cmd] ', b'')

#Считывание зашифрованного сообщения с сервера

proc.sendline(b'read')

data = proc.recvuntil(b'[cmd] ')

data = data.replace(b'[shhh] ', b'')

c\_base64 = data.replace(b'\n[cmd] ', b'')

#Инициализация переменных

n = bytes\_to\_long(base64.b64decode(n\_base64))

print('n = ', n)

e = 65537

print('e = ', e)

c = bytes\_to\_long(base64.b64decode(c\_base64))

print('cypher = ', c)

print('Инициализация завершена, начинаем поиск достаточно длинного s,

чтобы выполнилось s\*m>n. Подождите.-------------')

s = 0x81

count = 2

while True:

s += pow(16, count) \* 15

c1 = pow(s, e, n) \* c

valid = orcale\_ifValid(c1)

if not valid:

break

count += 1

print(hex(s))

print('Длина подобрана, начинаем уменьшать значение s,

чтобы минимизировать r. Это займет какое-то время...-----------')

times = 0

while True:

if times < 15:

s -= pow(16, count)

c1 = pow(s, e, n) \* c

if orcale\_ifValid(c1):

s += pow(16, count)

times = 0

count -= 1

if count == 1:

break

else:

times += 1

else:

times = 0

count -= 1

if count == 1:

break

print(hex(s))