**ЛАБОРАТОРНА РОБОТА № 5**

В цьому завданні вам слід розібратися з атакою за рахунок оракула та заповнювача (padding oracle attack) . Атакувати слід навчальний сайт за наступною адресою crypto-class.appspot.com. Уразливість, яка виникає за рахунок оракула та заповнювача, існує в багатьох продуктах, наприклад у пристроях безпеки, які мають назву secure tokens. Дане завдання повинно продемонструвати як можна використати цю уразливість. При бажанні можна ознайомитись детальніше з цією уразливістю та атаками, що її використовують, за посиланням: Vaudenay's paper.

Як реалізувати атаку. Припустимо зловмисник хоче викрасти секретну інформацію з цільового веб сайту crypto-class.appspot.com. Атакуючий підозрює, що цей веб сайт передає зашифровані дані користувача в параметрах URL наступним чином:

http://crypto-class.appspot.com/po?er=f20bdba6ff29eed7b046d1df9fb7000058b1ffb4210a580f748b4ac714c001bd4a61044426fb515dad3f21f18aa577c0bdf302936266926ff37dbf7035d5eeb4

Тобто, коли користувач Аліса взаємодіє з сайтом, сайт вміщує подібну URL у веб сторінки, які пересилаються Алісі. Атакуючий перехопив URL, яка наведена вище, та здогадався, що шифротекст, який слідує після "po?er=" це шістнадцяткове представлення шифротексту, який отриманий за допомогою AES CBC з випадковим вектором IV. Шифротекст містить деяку конфіденційну інформацію про сесію Аліси.

Після деяких експериментів, атакуючий з’ясовує, що цей веб сайт уразливий до атаки за рахунок CBC, оракула та заповнювача (CBC padding oracle attack). Зокрема, коли розшифрований в режимі CBC шифротекст завершується невірним заповнювачем, то веб сервер повертає код помилки 403 (forbidden request). З іншого боку, коли CBC заповнювач вірний, але повідомлення не відповідає формі, то сервер повертає код помилки 404 (URL not found).

Озброївшись цією інформацією, ви повинні розшифрувати вищезазначений шифротекст. Щоб це зробити вам слід надсилати HTTP запити до веб сайту у формі

http://crypto-class.appspot.com/po?er="your ciphertext here"

і спостерігати коди помилок, які повертаються.

Такі запити до веб сайту нагадують спілкування з оракулом. Це пояснює назву атаки. Таке «спілкування» з оракулом дозволяє розшифровувати повідомлення побайтно. Щоб визначити один байт слід надіслати 256 HTTP запитів до сайту. Майте на увазі, що перший блок шифротексту це випадковий вектор IV, а повідомлення, яке дешифрується, зберігається у ASCII кодуванні.

**ВИХІДНИЙ КОД**

package main

import (

"encoding/hex"

"fmt"

"net/http"

"strconv"

"strings"

)

func DecodeHex(key string) string {

decoded, err := hex.DecodeString(key)

if err != nil {

fmt.Println("Errorr")

}

return string(decoded)

}

func MakeRequest(url string) int {

resp, err := http.Get(url)

fmt.Println(url)

if err != nil {

// handle error

}

defer resp.Body.Close()

//fmt.Println(resp.StatusCode)

if resp.StatusCode == 404 {

return 1

}

return 0

}

func MakeCharsArray() []int {

var char\_array []int = []int{}

char\_array = append(char\_array, 32)

for start := 97; start < 123; start++ {

char\_array = append(char\_array, start)

}

for start := 65; start < 91; start++ {

char\_array = append(char\_array, start)

}

return char\_array

}

func ConvertStringToArray(text string) []string {

var char\_array []string = []string{}

for i := range text {

char\_array = append(char\_array, string(text[i]))

}

return char\_array

}

func CustomConverter(text string) []string {

var char\_array []string = []string{}

for i := 0; i < len(text); i += 2 {

ind := i

char\_array = append(char\_array, text[ind:ind+2])

}

return char\_array

}

func Reverse(s string) (ret string) {

for \_, v := range s {

defer func(r rune) { ret += string(r) }(v)

}

return

}

func PaddingOracleAttack(iv string, ct string) string {

iva := CustomConverter(iv)

iv\_index := len(iva) - 1

j := 0x1

//fmt.Println(iv\_index)

bf\_range := MakeCharsArray()

for k := range make([]int, 16) {

fmt.Println("k=", k)

temp := iva[iv\_index]

for i := range bf\_range {

num, e := strconv.ParseInt(iva[iv\_index], 16, 64)

if e != nil {

panic(e)

}

nxor := int(num) ^ bf\_range[i] ^ j

xor := strconv.FormatInt(int64(nxor), 16)

if e != nil {

panic(e)

}

iva[iv\_index] = xor

url := base\_url + "" + strings.Join(iva, "") + ct

//fmt.Println(nxor)

result := MakeRequest(url)

if result == 1 {

plain\_text += string(bf\_range[i])

break

}

iva[iv\_index] = temp

}

iv\_index -= 1

j += 1

list\_index := 0

end\_index := len(iva) - 1

for end\_index > iv\_index && list\_index < len(plain\_text) {

if k == 15 {

break

}

ord, e := strconv.ParseInt(iva[end\_index], 16, 32)

if e != nil {

panic(e)

}

iva[end\_index] = strconv.FormatInt(int64(int(ord)^j^(j-1)), 16)

end\_index -= 1

list\_index += 1

}

}

return plain\_text

}

var base\_url string = "http://crypto-class.appspot.com/po?er="

var plain\_text string = ""

func main() {

//fmt.Println(ConvertStringToArray("sdfsdfsdf"))

cipher\_text := "f20bdba6ff29eed7b046d1df9fb7000058b1ffb4210a580f748b4ac714c001bd4a61044426fb515dad3f21f18aa577c0bdf302936266926ff37dbf7035d5eeb4"

iv := cipher\_text[:32]

first\_block := cipher\_text[32:64]

fmt.Println("PR ", Reverse(PaddingOracleAttack(iv, first\_block)))

**РЕЗУЛЬТАТ ВИКОНАННЯ**

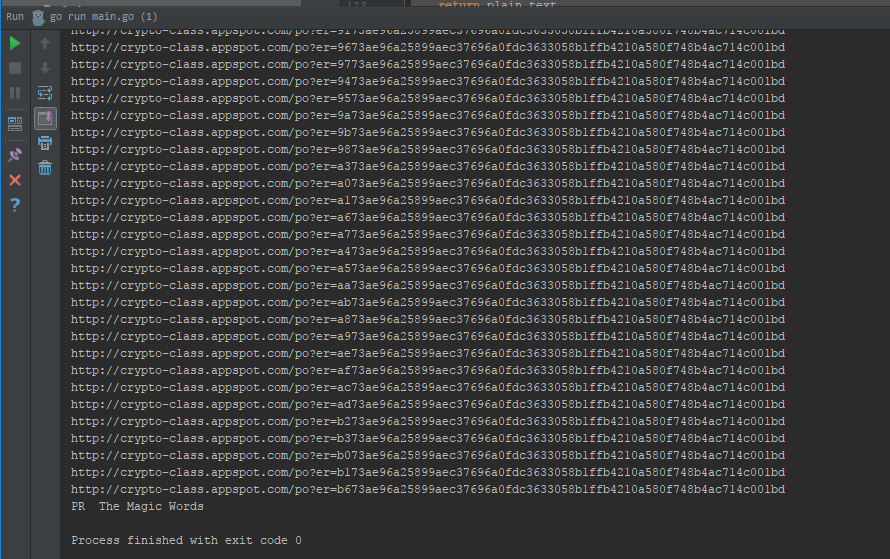


Рисунок 1 – Результат виконання