

Seccon CTF 2021 writeups

r10921a03 張承遠

Web

Vulnerabilities

Observation

1. 這題給了一個Go的後端code, 裡面有一個GET API跟一個POST API
2. Flag藏在DB的某個row裡面, 但在網頁上不會顯示出來
3. POST API會去query database, query用的參數為request的body, 且body內name的key不得為空。

Exploitation

這題由於Get API無法傳任何參數進去, 也不會使用任何參數, 因此把解題重點放在POST API上面。由於不知道Flag那個row的任何資訊(row裡面每個column), 因此要想辦法leak出來。

一開始是朝sql injection的方向去想, 然而這題使用參數化查詢, 因此會把傳進來的值當成是一般的字串使用, 查了這題使用到的gorm文件後發現很難去做sql injection。

後來比賽結束後發現真的不是sql injection, 而是此題用來存request body的struct有漏洞。在struct中有一個gorm.Model的struct, 裡面有包含了id, created time, update time等資訊, 因此當我們在傳request時在body加入id, 便有可能可以搜索到Flag那行row。

```
type Vulnerability struct {
    gorm.Model
    Name string
    Logo string
    URL  string
}
```

(程式裡用來存request body的struct)

```
// gorm.Model definition
type Model struct {
    ID          uint           `gorm:"primaryKey"`
    CreatedAt   time.Time
    UpdatedAt   time.Time
    DeletedAt   gorm.DeletedAt `gorm:"index"`
}
```

(gorm.Model 的詳細資訊)

接著還有一個問題要解決，因為POST API的request body有規定必須包含Name，但我們並不知道Name的值。這裡可以利用 gorm 當query的值為 "" 時會忽略該參數的特性，以及sqlite的column name是case insensitive的特性，在Body依序 Name="test", name="", id=14，這樣當gorm在處理query時，會過json["Name"] != null的檢查，且由於Name及name在sqlite3被視為是同一個，因此後面的會蓋掉前面的(name="" 會蓋掉 Name="test")，且gorm處理query會忽略空字串查詢，因此整個query實際上就剩下 id=14，也就順利拿到Flag。

MISC

s/<script>/gi

Observation

1. 這題給了一串很長的字串，並且要我們把所有<script>都移除，如果移除後會產生新的<script>也要移除，直到沒有任何的<script>，就可以拿到Flag

Exploitation

這題給的字串很長，因此必須要想到一個好的演算法去解析這個字串，如果每次暴力搜尋所有<script>並刪除，接著繼續搜尋刪除過後的字串有沒有<script>會花很久時間，時間複雜度約為O(n^2)。

後來有想到一個O(n)的演算法，創一個stack並從開始掃到結尾，每次把讀到的字元放進stack裡面，並且每次檢查stack裡面的前8個是不是<script>，若是的話就把這8個從stack中移除。因為移除過後的stack不可能包含<script>（有的話前面應該已經pop掉了），因此就不需要跑多餘的重複運算，最後成功拿到Flag。