

2023/03/20

入力検証

菅田大輔

目次

- 入力検証とは
- 入力検証の方法
- OWASP Juice Shopにおける実践
- Payback Time
- 追記：メールアドレスの検証

入力検証とは

- 入力検証とは、情報システムにデータが正しく入力されていることを確認するプロセスである。これにより、データベースに誤ったデータが保存されるのを防ぎ、システムの誤作動を回避できる。
- 入力検証は重要であるが、完全なセキュリティ対策ではない。攻撃者は予想しない方法でシステムを悪用する可能性があるため、他のセキュリティ対策（例：パラメータ化クエリ、アクセス制御）等も併用することが重要である。

入力検証の方法

- 入力検証は、**構文レベル**と**セマンティックレベル**の両方で適用する必要があります。

- **構文**検証

構造化フィールドの正しい構文を適用する必要があります
(例：SSN、日付、通貨記号)。

- **セマンティック**検証

特定のビジネスコンテキストで値の正確性を強制する必要があります（たとえば、開始日が終了日より前、価格は予想範囲内です）。

OWASP Juice Shopにおける実践

- Payback Time: Place an order that makes you rich.

- 概要

入力検証の脆弱性を利用し、ウォレットの残高を増やすようなオーダーを行う。Juice-shop側で、入力された値に対するセマンティック検証がなされていないため、商品の個数が負の値でも受け付けてしまう。これを利用して、個数が負の値の商品の決済を行い、ウォレットの残高を増やす。

Payback Time

PUTリクエスト：PUTリクエストは、リクエストされたリソースがすでに存在する場合は更新し、存在しない場合は新規に作成するリクエストのこと。

1. 方針として、まず実際に商品購入を行い、Networkログを観察する。
2. バスケット内のアイテム数を変更すると、
<http://localhost:42000/api/BasketItems/11>へPUTリクエストが飛んでいることがわかる。
3. ソースコードのupdateNumberOfCartItemsを見てみると、BasketItem.quantityがバスケット内の商品の個数を変更していることがわかる。

```
updateNumberOfCartItems() {  
  this.find(parseInt(sessionStorage.getItem('bid'), 10)).subscribe(e=>{  
    this.itemTotal.next(e.Products.reduce((n, i) => n + i.BasketItem.quantity, 0))  
  }, e=>console.log(e))  
}
```

Payback Time

4.Quantityを負の値としたPUTリクエストを作成し、送信する。この時、Authorizationヘッダ、Content-Typeをこれまでのリクエストに従って記述する。

5.実際に商品購入の手続きを完了させる。

Authorizationヘッダ:
一般に、以下のような形式を持つ
<type> <credentials>

The screenshot shows a REST client interface with the following configuration:

- Method:** POST
- URL:** http://localhost:42000/api/BasketItems/11
- URL Parameters:** None
- Headers:**
 - ☒ Authorization: Do0Mjo0Ni44NTIgKzAwOjAwliwiZGVsZXRLZEF0IjpudWxsfsWiaWF0IjoxNjc5MjY3MjE3LCJleHAiOiE2NzkyODUyMTd9.NCAsBBYi4AkCXV9yviWf_DYZFZnYEOxgEbi6AtcJiu2OYYumFe20oWOM5Se08be7Kqy-KJLiGr8UkFhgjk1ebTQPfPuqljX_L4meGHvFvjP4kPmJGR85Z2RBLVgM0RkpspQPHAUAo_keG17ps6yvtC3JCxK0olT6RFI5Jkx2QS8
 - ☒ Content-Type: application/json
 - ☒ name: value
- Body:** {"quantity": -100}

At the bottom right, there are "Clear" and "Send" buttons.

Payback Time

- 結論

値に対するセマンティック検証が行われていないという脆弱性が存在する。これを利用して、ショッピングバックの商品の個数を負の値にし、決済を行うことでユーザー側が一方的に得をするオーダーを行うことができる。

Payback Time

- 想定される被害

想定していない個数の商品がユーザーが選択できてしまうことで、不利益を被ったり、情報が漏洩する恐れがある。

- 対策

数値パラメータの最小値と最大値の範囲チェックを行う。（=セマンティック検証）

RFC 5321:RFC 5321は、Simple Mail Transfer Protocol (SMTP)の標準仕様を定めた文書。SMTPは、電子メールを送信するためのプロトコル。

プロトコル：相互に通信するための規約や手順

追記：メールアドレスの検証

- 構文検証

電子メールアドレスの形式はRFC 5321で定義されており、かなり複雑なアドレスも存在可能である。

- " "@example.org

しかし、実際のメールサーバーは限定されたアドレスしか扱っていない。そのため、基本的な初期検証を実行し、アドレスをメールサーバーに渡し、拒否した場合に例外をキャッチすることが対策になる。

有効な初期検証として、以下のものが挙げられる。

- メールアドレスに@記号で区切られた2つの部分が含まれている。
 - メールアドレスに危険な文字(“”、’、`、など)は含まれてない。

追記：メールアドレスの検証

- セマンティック検証

最も一般的な方法は、ユーザーに電子メールを送信し、電子メール内のリンクをクリックするか、送信されたコードを入力すること。これは、次のような基本的なレベルの保証をする。

- 正しいメールアドレスであること。
- アプリケーションは正常にメールを送信できること。
- ユーザーがメールボックスにアクセスできること。