2022 年 08 月 25 日

サービス＆セキュリティ株式会社 e-Gate センター

Web アプリケーションにおける入力値検証について

1. 概要

一般的な Web アプリケーションにおいて入力値検証は実装されていることが多いものの、その実装の不備により生じる脆弱性は弊社の脆弱性診断サービスでも頻繁に検出されています。

本記事では、Web アプリケーションの入力値検証について、実装場所によるセキュリティ対策の有効性の違いを主に解説します。Web アプリケーションのセキュリティ対策の一助になれば幸いです。

2. 入力値検証(バリデーション)とは

一般的な Web アプリケーションには、クライアント側の入力に対してサーバ側でそれに応じた動作を行う機能が様々実装されています。たとえば、検索機能の場合、クライアントは検索したい値を入力すると、サーバがデータベースに問合せ、その結果をクライアント側に出力します。

ここで、検索機能が、「郵便番号を入力し、それに対応する住所を検索する」という機能である場合、Web アプリケーションの開発者は、正常な動作を実現するため、入力値が 7 桁の数字であるかを確かめる処理を実装します。このような入力値が期待している形式であるかどうかを確かめる処理を、入力値検証(バリデーション)と呼びます。

入力値検証の主な目的は Web アプリケーションを仕様通りに動作させることですが、セキュリティ対策としての効果も併せて期待されている場合もあります。つまり、悪意ある者によって送信される攻撃コードの実行を入力値検証することで防止するという効果です。

入力値検証は実装する場所により大きく 2 つに分けられます。1 つが HTML や JavaScript といった**クライアント側**で動作するコード内での実装であり、もう 1 つが PHP や Ruby など**サーバ側**で動作するコード内での実装です。次章以降でその違いを見ていきます。

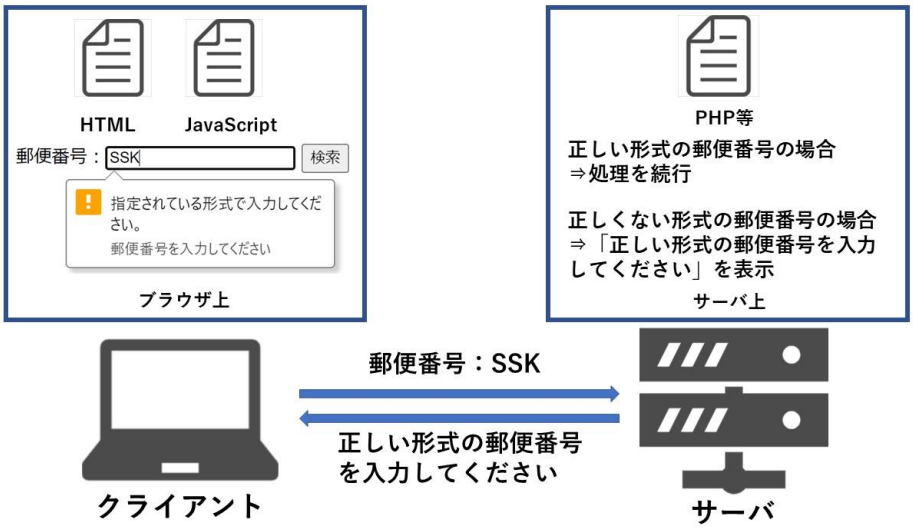

図 1:入力値検証の概略

3. クライアント側の入力値検証

まず、クライアント側での入力値検証の実装について説明します。

クライアント側での入力値検証は HTML または JavaScript によって実現することができます。以下は、郵便番号のみを許可する入力値検証を実装した HTML の一部抜粋です。

```
<form method="GET" action="return_address.php">
    <label>郵便番号：<input type="text" name="zipcode" pattern="[0-9]{3}-?[0-9]{4}"
title="郵便番号を入力してください"></label>
    <button>検索</button>
</form>
```

この HTML をブラウザ(今回は Google Chrome)で表示してみます。

郵便番号の形式ではない文字を入力して検索ボタンを押すと、図 2 のようにエラーが表示され、送信を抑止します。これは、input 要素の pattern 属性にて、郵便番号の形式を正規表現で設定することで実現しています。これにより利用者は、フォームへの入力時あるいは送信する前などに、入力した値が想定されていない形式であることを知ることができ、ユーザエクスペリエンス(UX)の向上につながります。

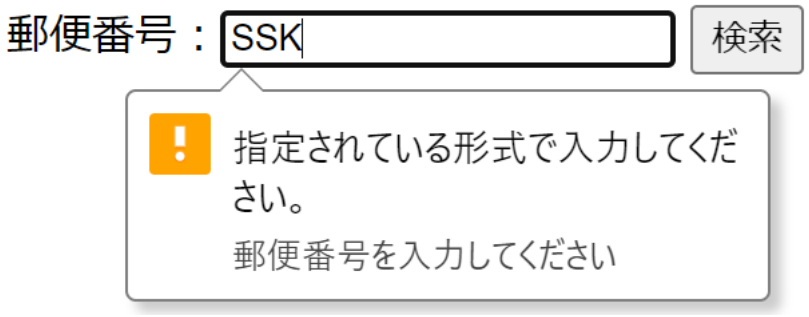

図 2:クライアント側の入力値検証

UX の向上にも繋がるクライアント側の入力値検証ですが、セキュリティ面の効果もあるのでしょうか。

一見、クライアント側から郵便番号以外の値をサーバ側に送信することはできないように見えるため、悪意ある者による攻撃コードの送信は不可能のように思えます。しかし、実は簡単な手順で、この入力値検証を迂回して任意の値を送信することができます。

その方法の 1 つとして、ブラウザに備わっている開発者ツールを用いて、HTML を書き換えることによる迂回があります。

まず、ブラウザにて「F12」キーを押すなどして、開発者ツール(Google Chrome の場合、DevTool という名称)を起動します。次に、「Elements」タブを開き、先ほど示した郵便番号の入力箇所(input 要素)を探します。

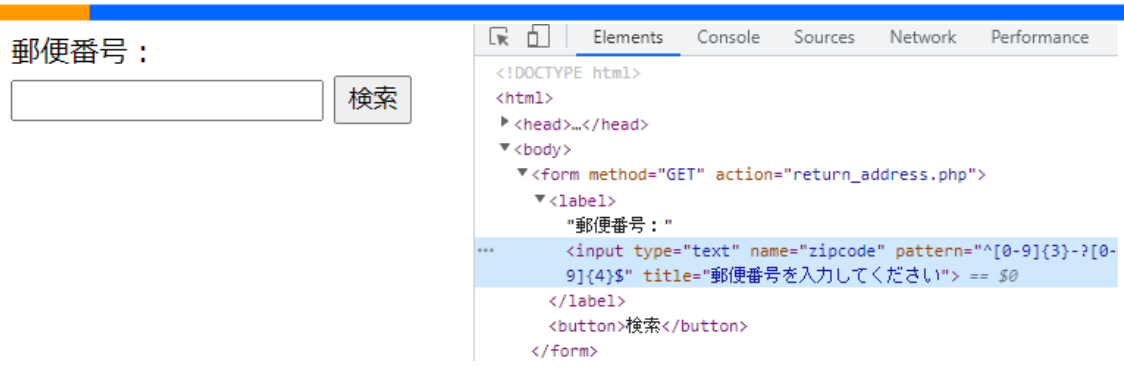

図 3:開発者ツールにて郵便番号の入力箇所(input 要素)の表示

ここで、「pattern="[0-9]{3}-?[0-9]{4}"」の部分を削除します。すると、適当な文字を入力して検索ボタンを押してもエラーが発生せず、サーバ側にデータを送信できてしまいます。

図 4: 開発者ツールにて入力値検証処理の削除後

このように、HTML をはじめとするクライアント側のコードによる入力値検証は容易に回避可能です。そのため、Web アプリケーション開発者は、クライアント側のコードによる入力値検証をセキュリティ面においては信頼してはいけません。

4. サーバ側の入力値検証

次に、サーバ側での入力値検証の実装について説明します。

3 章で使用した HTML にて、検索ボタンを押すと、サーバ側の「return_address.php」へ郵便番号を送信し、サーバ側ではデータベースから対応する住所を検索します。

以下は、入力値検証を実装した return_address.php の一部抜粋です。

```
<?php
    $zipcode = $_GET["zipcode"];
    if (!preg_match("/^[0-9]{3}-?[0-9]{4}$/", $zipcode)){
        echo "正しい形式の郵便番号を入力してください";
    }
    else{
        $sql = "SELECT addr FROM addr_list WHERE zipcode = $zipcode";
        $result = mysql_query($sql);
```

上記の PHP コードは、if 文によって、正しい郵便番号の形式でない場合に「正しい形式の郵便番号を入力してください」というエラーを出力します。クライアント側の入力値検証が破られた場合でも、サーバ側で入力値検証を行い、Web アプリケーションの意図した動作を保証します。サーバ側のコードは、サーバ内に侵入しない限り書き換えることはできないため、クライアント側の入力値検証とは異なり、確実な入力値検証となります。

一方、上記の PHP コードにおいて、入力値検証を実装していない場合を考えます。

```
<?php
    $zipcode = $_GET["zipcode"];
    $sql = "SELECT addr FROM addr_list WHERE zipcode = $zipcode";
    $result = mysql_query($sql);
```

上に示した 2 つの PHP コードは共に、SQL 文の組み立てに単純な文字列結合を使用しています。1 つ目の PHP コードでは入力値検証が実装されており、郵便番号として適切な形式でないと SQL クエリが発行されないため、結果的に SQL インジェクションを回避することができています。しかし、2 つ目の PHP コードでは、SQL インジェクションの脆弱性があります(参考資料[1][2])。郵便番号として SQL インジェクションの攻撃パターンを入力することで、データベース内のデータを削除されたり、データを抜き取られたりする可能性があります。

ここまで SQL インジェクションを例として入力値検証の重要性について説明しましたが、SQL インジェクションの**根本的対策**は、SQL 文の組み立てにプレースホルダを使用することです。そのため、たとえサーバ側に入力値検証が実装されてなくとも、プレースホルダを使用して SQL 文を正しく組み立てることで原理的に SQL インジェクションの脆弱性は入り込みません。

今回の郵便番号のような単純な形式であれば、入力値検証によって結果的に SQL インジェクション脆弱性の対策となりました。しかし、入力値検証のみ行い根本的対策であるプレースホルダの使用を怠ると、入力値が複雑な場合に検証をすり抜ける攻撃パターンが残存したり、検証の条件が複雑で実装に不備がある場合に脆弱性が生じたりします。

とはいえ、サーバ側での入力値検証はセキュリティ対策において不要とは言えません。それには以下のような理由が挙げられます。

- 根本的対策が実装不備などの理由で有効でない場合に、**保険的対策**となり得る場合があるため
- 入力値検証が根本的対策となり得る脆弱性も存在するため

脆弱性診断の結果、脆弱性が検出されなかった場合においても WAF(Web Application Firewall)の導入を推奨するように、セキュリティは**多層防御**が重要です。脆弱性に対する根本的対策に加えて、保険的対策も併用することで堅牢な Web アプリケーションシステムを構築することができます。

5. まとめ

本記事では、セキュリティ面における入力値検証について以下の点を示しました。

- クライアント側の入力値検証はセキュリティ面において信頼することはできない
- サーバ側の入力値検証は脆弱性への根本的対策になり得ない場合もあるが、その場合においても保険的対策となり得る場合がある

弊社の脆弱性診断においても、クライアント側の入力値検証を信頼し、サーバ側で対策を行わなかった結果生じたであろう脆弱性を検出することが度々あります。Web アプリケーションを開発、運用していて、クライアント側の入力値検証に頼っていないか等のセキュリティ面において不安がある場合、是非弊社の脆弱性診断をご検討ください。

6. e-Gate の監視サービスおよび脆弱性診断サービスについて

e-Gate のセキュリティ機器運用監視サービスでは、24 時間 365 日、リアルタイムでセキュリティログの有人監視を行っております。サイバー攻撃への対策としてセキュリティ機器を導入する場合、それらの機器の運用監視を行い、通信が攻撃かどうかの分析、判断をして、セキュリティインシデント発生時に適切に対処できるようにすることが重要です。e-Gate のセキュリティ監視サービスをご活用いただきますと、迅速なセキュリティインシデント対応が可能となります。

また、e-Gate の脆弱性診断サービスでは、お客様のシステムを診断し、検出された脆弱性への対策をご提案させていただいております。テレワークの常態化や IoT 等のデバイスの多様化が進む昨今、特定の攻撃経路だけを想定した「境界防御」に加えて、脆弱性を把握・管理・対処する『本質防御』も必須となっています。e-Gate の脆弱性診断サービスをご活用いただきますと、お客様のシステムにおける脆弱性の存否が明らかになります。

監視サービスや脆弱性診断サービスをご活用いただきますと、セキュリティインシデントの発生を予防、また発生時にも迅速な対処が可能なため、対策コストや被害を抑えることができます。

■総合セキュリティサービス e-Gate

SSK（サービス＆セキュリティ株式会社）が 40 年以上に渡って築き上げてきた「IT 運用のノウハウ」と最新のメソッドで構築した「次世代 SOC “e-Gate センター”」。この 2 つを融合させることによりお客様の情報セキュリティ全体をトータルにサポートするのが SSK の“e-Gate”サービスです。e-Gate センターを核として人材・運用監視・対策支援という 3 つのサービスを軸に全方位のセキュリティサービスを展開しています。

【参考 URL】

https://www.ssk-kan.co.jp/e-gate

7. 参考資料

[1] IPA,「安全なウェブサイトの作り方」改訂第 7 版, 2022 年 8 月 15 日閲覧
https://www.ipa.go.jp/files/000017316.pdf

[2] IPA, 別冊：「安全な SQL の呼び出し方」, 2022 年 8 月 15 日閲覧
https://www.ipa.go.jp/files/000017320.pdf

※本資料には弊社が管理しない第三者サイトへのリンクが含まれています。各サイトの掲げる使用条件に従ってご利用ください。
　リンク先のコンテンツは予告なく、変更、削除される場合があります。
※掲載した会社名、システム名、製品名は一般に各社の登録商標 または商標です。

«お問合せ先»
サービス＆セキュリティ株式会社
〒150-0011
東京都渋谷区東 3 丁目 14 番 15 号 MO ビル 2F
TEL 03-3499-2077
FAX 03-5464-9977
sales@ssk-kan.co.jp