バリデーションに関する機能の作成に対して取り組んだ内容

このシステムでは、外部から受け取ったデータに関しては、不正なデータを一切に受け付けないようにバリデーションを設けています。

この資料の目的は、セキュリティを確保するうえで気を付けた点や実装内容をプレゼンします。

目次

バリデーションの対象とポリシー

このシステムでは、以下の処理をバリデーションの対象としています。

ユーザーから入力されたデータ

XSSやSQLインジェクション攻撃等を防ぐためには、必要不可欠です。

システム側からユーザーに対して送信するデータ

このシステムでは、出力時のデータも対象としています。出力しようとしているデータが改ざんされていたり、
そもそも入力時のバリデーションに不具合があって不正なデータが存在した際、誤ってユーザーに送ってしまい
XSSが起こってしまうことを防ぐためです。

なお、バリデーション時の評価方法は、レスポンスを早めるため短絡評価としています。

ユーザーからのデータのエラー項目の通知は、フロントエンド側に任せて対処する事にしており、
バックエンド側である本システムは「入力データの最終チェック」という目的で行っています。

そのため、厳密にどの項目でエラーが発生したのかをバックエンド側が判定して返す必要はない為、
「一つでも不正が見つかれば、即エラーとしてレスポンスを返す」事にしています。

バリデーションを行うデータ

バリデーションを行うデータとしては、以下の通りです。

フォームからの文字列データ

具体的には、XSSの危険性があるような文字（例：<>:;_.など）が含まれていないか確認したり、データベースが格納できるような文字数になっている事、あらかじめ指定した正規表現に合致することなどを判定します。

フォームからのバイナリデータ

バイナリデータのMIMEタイプや拡張子が条件に合致している事や、データ量が超過していないかを確認したり
ファイル名にXSSやディレクトリトラバーサル攻撃の危険性がある文字列が含まれていないことを判定します。

CSVファイルから抽出した文字列データ

フォームからの文字列データを判定する際の条件と同じ条件で、抽出した文字列を判定します。
なお、レスポンス時には、エラーが発生したCSVファイルの表内の行番号と列番号を返却します。

ZIPファイルから抽出したバイナリデータ

フォームからのバイナリデータを判定する際の条件と同じ条件で、抽出したバイナリデータを判定します。

これらの判定を、入力時と出力時の両方に行います。

文字列のバリデーション時に気を付けたこと

文字列のバリデーション時には以下の事を徹底しました。

ブラックリスト判定ではなく、ホワイトリスト判定で行う

ブラックリスト判定だと、危険な対象の定義漏れなどでセキュリティホールが発生する恐れがあります。
若干利便性は落ちますが、ホワイトリスト判定を行うことで安全にバリデーションが可能になります。
そのため、実装してある正規表現文字列は、全てホワイトリスト判定で組んであります。

文字列は、要件上特に問題がなければ「全角文字」で受け付ける

XSSなどの危険性がある文字は、すべて半角文字です。そのため、全角文字しか受け付けないようにしておけば、
利便性は下がりますが危険性をかなり抑えることができます。

サニタイズは、可能な限りフレームワークや外部ライブラリの関数に任せる

Javaの標準の機能に、残念ながらPHPみたいにサニタイズ用の便利な関数は存在しません。正規表現で
まかなうしかありません。ですが、それではサニタイズに漏れが生じてしまい、極めて危険です。
サニタイズ含めバリデーションに関わる機能は、可能な限り外部プログラムに頼ることでリスクを軽減できます。

そもそも、サニタイズ対象の文字列は受け付けない

このシステムでは、サニタイズ対象の文字列をサニタイズして用いるのではなく、「そもそも少しでも含まれて
いたらエラーとして、処理を行わない」方針にしています。

バイナリデータの検査時に気を付けたこと

文字列のバリデーション時には以下の事を徹底しました。

ファイル名に、ファイルパス構成文字やヌル文字がないかを確認する

この二つの文字列がファイル名に存在すると、ディレクトリトラバーサル攻撃のリスクが非常に高まります。
そのため、例えば「/」「.」「¥」といったファイルパス構成文字や、「¥0」といったヌル文字の存在を確認し発見したら即エラーとしています。なお、サニタイズ対象文字も判別しています。

ファイルの種類は、拡張子だけでは判別しない

ファイル名についている拡張子は、いくらでも偽装が可能なので、拡張子だけでファイルの種類を判別すると不正な実行ファイルを受け付けてしまう恐れがあり、極めて危険です。
このシステムでは、「Apache Tika」というライブラリを用いて、バイト配列から直接読み取ってファイル種別を判定するようにしています。
バイト配列からの判定結果と、拡張子からの判定結果の辻褄が合わなければエラーです。

絶対に、ファイルのデータ量を確認する

必ず確認しておかないと、不正に大きすぎるデータを渡されて、バッファオーバーフロー攻撃の原因となる場合がありますので、必須項目です。

CSVファイルのバリデーション時の工夫

CSVファイルによるデータの一括投入の際は、バリデーションエラーがたくさん出ることが予想されます。
ですが、このCSVファイルバリデーションの結果まで短絡評価としてしまうと、ユーザーがCSVファイル内のどの部分を修正すればよいのか分からず、大変困ってしまいます。

そこでエラー時には、エラーが発生したCSVファイルの行番号と列番号をリストで返却するように実装しております。
これによって、修正が楽になります。

（例）CSVファイルの以下の部分にエラーが発生した場合

管理番号	名前	ふりがな	担当楽器
abc-123	森本	もりもと	トランペット
efg-456	伊藤	いとう	ヴァイオリン
hij-789	佐藤	さとう	フルート

以下のような二重配列を返却し、
・2行目の1列目と3列目
・4行目の4列目
にエラーがあることを教えてくれます。

```
2: [1, 3]
4: [4]
```