Input Validation Cheat Sheet

From OWASP

Introduction

This article is focused on providing clear, simple, actionable guidance for providing Input Validation security functionality in your applications.

White List Input Validation

It is always recommended to prevent attacks as early as possible in the processing of the user's (attacker's) request. Input validation can be used to detect unauthorized input before it is processed by the application. Developers frequently perform black list validation in order to try to detect attack characters and patterns like the ' character, the string 1=1, or the <script> tag, but this is a massively flawed approach as it is typically trivial for an attacker to avoid getting caught by such filters. Plus, such filters frequently prevent authorized input, like O'Brian, when the ' character is being filtered out.

White list validation is appropriate for all input fields provided by the user. White list validation involves defining exactly what IS authorized, and by definition, everything else is not authorized. If it's well structured data, like dates, social security numbers, zip codes, e-mail addresses, etc. then the developer should be able to define a very strong validation pattern, usually based on regular expressions, for validating such input. If the input field comes from a fixed set of options, like a drop down list or radio buttons, then the input needs to match exactly one of the values offered to the user in the first place. The most difficult fields to validate are so called 'free text' fields, like blog entries. However, even those types of fields can be validated to some degree, you can at least exclude all non-printable characters, and define a maximum size for the input field.

Developing regular expressions can be complicated, and is well beyond the scope of this cheat sheet. There are lots of resources on the internet about how to write regular expressions, including: http://www.regular-expressions.info/. The following provides a few examples of 'white list' style regular expressions:

```
White List Regex Examples
Validating Data from Free Form Text Field for Zip Code (5 digits plus optional -4) ^\d{5}(-\d{4})?$

Validating Data from Fixed List Drop-Down Box For U.S. State Selection
```

```
  ^(AA|AE|AP|AL|AK|AS|AZ|AR|CA|CO|CT|DE|DC|FM|FL|GA|GU|HI|ID|IL|IN|IA|KS|KY|LA|ME|MH|MD|MA|MI|MN|MS|
  MO|MT|NE|NV|NH|NJ|NM|NY|NC|ND|MP|OH|OK|OR|PW|PA|PR|RI|SC|SD|TN|TX|UT|VT|VI|VA|WA|WV|WI|WY)$

Validating a Free Form Text Field for allowed chars (numbers, letters, whitespace, .-_)
  ^[a-zA-Z0-9\s\.\-_]+$         (Any number of characters)
  ^[a-zA-Z0-9\s\.\-_]{1-100}$   (This is better, since it limits this field to 1 to 100 characters)
Note: \s matches any whitespace character (i.e., space, tab, carriage return, or linefeed, [ \t\r\n])

Additional Examples are available at the OWASP Validation Regex Repository
```

Java Regex Usage Example

```
Example validating the parameter “zip” using a regular expression.

private static final Pattern zipPattern = Pattern.compile("^\d{5}(-\d{4})?$");
public void doPost( HttpServletRequest request, HttpServletResponse response) {
       try {
               String zipCode = request.getParameter( "zip" );
               if ( !zipPattern.matcher( zipCode ).matches()  {
                       throw new YourValidationException( "Improper zipcode format." );
               }
               .. do what you want here, after its been validated ..
       } catch(YourValidationException e ) {
               response.sendError( response.SC_BAD_REQUEST, e.getMessage() );
       }
}
```

Some white list validators have also been predefined in various open source packages that you can leverage. Two packages that provide this are:

- Apache Commons Validator (http://jakarta.apache.org/commons/validator)
- OWASP ESAPI Validators

It is strongly recommended that you use ESAPI to assist with your input validation needs, rather than writing your own validation routines. The OWASP Enterprise Security API (ESAPI) project has predefined validators defined in the org.owasp.esapi.Validator interface and implemented in the DefaultValidator reference implementation. These include:

- getValidDate()
- getValidSafeHTML()
- getValidInput()
- getValidNumber()
- getValidFileName()
- getValidRedirectLocation()

With ESAPI, the previous example can be rewritten as follows:

```
Example validating the parameter “zip” with generic ESAPI input validator.

public void doPost( HttpServletRequest request, HttpServletResponse response) {
       try {
```

```
                String zipCode = Validator.getValidInput("ChangeAddressPage_ZipCodeField",
                  request.getParameter( "zip" ), "zipCodePattern", 10, false));
                .. do what you want with validated 'zipCode' param here ..
        } catch( ValidationException e ) {
                response.sendError( response.SC_BAD_REQUEST, e.getMessage() );
        }
}

// zipCodePattern is the name of a property defined in ESAPI.properties, and its value
// is the regular expression: "^\d{5}(-\d{4})?$"
//
// If zipcodes were a frequently used parameter in your application, we would recommend
// that you create your own getValidZipCode() method that builds on top of ESAPI, to make
// it even simpler for your developers to use.
```

- The overall javadoc for ESAPI is here (http://owasp-esapi-java.googlecode.com/svn/trunk_doc/index.html)
- And the javadoc for this specific interface is here (http://owasp-esapi-java.googlecode.com/svn/trunk_doc/org/owasp/esapi/Validator.html) .

Other Articles in the OWASP Cheat Sheet Series

- Authentication Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- Cryptographic Storage Cheat Sheet
- **Input Validation Cheat Sheet**
- DOM based XSS Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- HTML5 Security Cheat Sheet
- SQL Injection Prevention Cheat Sheet
- Security Architecture Cheat Sheet
- Session Management Cheat Sheet
- Transport Layer Protection Cheat Sheet
- XSS (Cross Site Scripting) Prevention Cheat Sheet
- Web Service Security Cheat Sheet

Authors and Primary Editors

Dave Wichers - dave.wichers [at] aspectsecurity.com