|  |  |
| --- | --- |
| February 18, 2011 |  |
|  | Security Framework |
| Version 1.0 |  |
| Decoders user’s guide | |



Table of Contents

1 Introduction 1

2 Configuration files structure 1

2.1 SubsystemProperties 1

2.2 Engine 2

2.2.1 Non-configurable engines 2

2.2.2 Entity decoder engine 3

2.2.3 Base 64 decoder engine 4

2.2.4 Complex Base 64 decoder engine 5

3 Decoder subsystem API 7

3.1 Simplified API 7

3.2 General API 9

3.3 API Examples 10

3.3.1 Simplified API 10

3.3.2 General API 11

# Introduction

Decoder subsystem provides tools of conversion from different common data formats, most of which can be used by malefactors to attack WEB sites. The functionality of subsystem covers the area where the .NET framework either doesn’t provide any standard tools or the standard tools are hard to use or they don’t work properly.

Decoder subsystem API is easy to use and its behavior can be changed via the configuration.

The subsystem includes a number of engines converting the data from HTML, XML, URL, JavaScript, VBScript, LDAP and CSS escaping, BASE64 and GZIP data formats.

# Configuration files structure

Configuration is a set of XML documents. Each Security Framework subsystem has a separate configuration file. There is a root configuration file - MtSfConfigurationLoader.xml that references all subsystem’s configuration files.

Please see the section “2.1. Security Monitor configuration reference” in the Security Monitor user’s guide for the root configuration file reference.

The Decoders configuration file structure at high level is shown below:

|  |
| --- |
| <SubsystemPropertiesIsRuntimeApiEnabled="true"IsRuntimeApiPublic="true"IsControlApiEnabled="true"IsControlApiPublic="true">  <Engines>  <Engine/>  <Engine/>  </Engines>  </SubsystemProperties> |

The following sections describe attributes, elements, and child elements.

## SubsystemProperties

Root element. It’s not required but recommended do not change the element’s name. It matches with name of the class that takes the configuration properties when it is loading.

There are the following child elements and attributes under the root.

Attributes:

| Attribute | Description |
| --- | --- |
| IsRuntimeApiEnabled | Required attribute.  Indicates whether the Decoder API is enabled. Prevents the API from initializing if set to **false**. |
| IsRuntimeApiPublic | Required attribute.  Indicates whether the Decoder API is exposed to public access. |
| IsControlApiEnabled | Required attribute.  Indicates whether the Decoder control API is enabled. Prevents the control API from initializing if set to **false**. *For future use*. |
| IsControlApiPublic | Required attribute.  Indicates whether the Decoder control API is exposed to public access. *For future use*. |

Elements:

| Element | Description |
| --- | --- |
| Engines | Required element.  Defines available engines and their properties. |

The element’s structure is described below.

## Engine

The Engine element’s structure depends on the engine, defined by the element. There are configurable and non-configurable engines. All non-configurable engines use the same configuration structure.

### Non-configurable engines

The following engines are non-configurable: URL, CSS, LDAP, JavaScript, VBScript and GZIP. The following attributes are available for them in the Engine element:

<SubsystemProperties>

<Engines>

|  |
| --- |
| <EngineId="<Engine\_Id>"IsDefault="true|false"RealType="MetraTech.SecurityFramework.EngineCommonProps" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Decoder API. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: the following engines are defined in the subsystem   * MetraTech.SecurityFramework.DefaultUrlDecoderEngine * MetraTech.SecurityFramework.Core.Decoder.JavaScriptDecoderEngine * MetraTech.SecurityFramework.Core.Decoder.VbScriptDecoderEngine * MetraTech.SecurityFramework.Core.Decoder.CssDecoderEngine * MetraTech.SecurityFramework.Core.Decoder.GZipDecoderEngine * MetraTech.SecurityFramework.Core.Decoder.LdapDecoderEngine |

### Entity decoder engine

The EntityDecoderEngine provides a possibility to decode from HTML and XML encodings. Its configuration described below.

<SubsystemProperties>

<EnginesProps>

|  |
| --- |
| <EngineId="<Engine\_Id>"Category="<Category\_Name>"IsDefault="true|false"  RealType="MetraTech.SecurityFramework.Core.Decoder.EntityDecoderEngine"  RequireTrailingSemicolon="true|false">  <NamedEntities>  <itemName="<Entity\_name>"Value="<Character>"/>  <itemName="<Entity\_name>"Value="<Character>"/>  </NamedEntities>  </Engine> |

Attributes:

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Decoder API. |
| Category | Required attribute.  An engine category.  The value must fit with the category defined in the engine. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Decoder.EntityDecoderEngine” |
| RequireTrailingSemicolon | Optional attribute.  Defines whether each entity must ends with semicolon (;) character.  Value: Boolean |

Elements:

| Element | Description |
| --- | --- |
| NamedEntities | Required element.  Defines a collection of named entities as a name-value pairs.  Each item in the collection has the following attributes:   * Name – specifies a name of the entity (the value if case sensitive). * Value – specifies a character represented by the entity in the form &Name; (where Name is value of Name attribute). |

### Base 64 decoder engine

Base 64 decoder provides decoding from different kinds of base 64 encoding.Its configuration described below.

<SubsystemProperties>

<EnginesProps>

|  |
| --- |
| <EngineId="<Engine\_Id>"IsDefault="true|false"RealType="MetraTech.SecurityFramework.Core.Decoder.Base64DecoderToStringEngine"CharForIndex62="<Character>"CharForIndex63="<Character>" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Decoder API. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Decoder.Base64DecoderToStringEngine” |
| CharForIndex62 | Optional attribute.  Specifies a character for encoding number 62.  Value: any single character (default ‘+’) |
| CharForIndex63 | Optional attribute.  Specifies a character for encoding number 63.  Value: any single character (default ‘/’) |

### Complex Base 64 decoder engine

Complex Base 64 decoder provides decoding from different kinds of base 64 encoding.On the input sequence decoder attempts to determine the version encoded as Base64. In case of successful identification of transfer required Base64 decoder.

This complex decoder used following Base64 decoders:

* + “Base64.Standart” - encoding for RFC 3548 or RFC 4648 (the char62 = '+' and the char63 = '/');
  + “Base64.ModifiedForFilenames” - modified Base64 for filenames (nonstandard)

(thechar62 = '+' andthe char63 = '-');

* + “Base64.ModifiedForXmlNmtoken” - modified Base64 for XML name tokens (Nmtoken)

(the char62 = '.' and the char63 = '-');

* + “Base64.ModifiedForXmlName” - modified Base64 for XML identifiers (Name)

(the char62 = '\_' and the char63 = ':');

* + “Base64. ModifiedForProgramIdentifiersV2” - modified Base64 for program identifiers(variant 1, nonstandard) (the char62 = '.' and the char63 = '\_');
  + “Base64.ModifiedForRegularExpressions” - modified Base64 for Regular expressions(nonstandard)

(the char62 = '!' and the char63 = '-');

Other versions of Base64 decoder can’t be determined.

Its configuration described below.

<SubsystemProperties>

<EnginesProps>

|  |
| --- |
| <EngineId="Base64.ComplexDecoder"IsDefault="false"RealType="MetraTech.SecurityFramework.Core.Decoder.ComplexBase64Decoder"/> |

# Decoder subsystem API

## Simplified API

Simplified Decoder subsystem API is provided by MetraTech.SecurityFramework.DecoderExtensions class and consists of extension methods for the System.String class.

To use the simplified API the reference to SecurityFramework.dll must be added and MetraTech.SecurityFramework must be declared in using directive.

The following methods are defined in the API:

| Method | Description |
| --- | --- |
| DecodeWithEngine(engineId: string): string | Invokes a decoder engine with the specified ID passing the string instance as an input.  Returns a decoding result (as a string).  Throws the SubsystemInputParamException when an engine with the specified ID not found.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromUrl(): string | Converts the string instance from URL encoding using a default URL decoder.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the URL category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromHtml(): string | Converts the string instance from HTML encoding using a default HTML decoder.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the HTML category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromHtmlAttribute(): string | Converts the string instance from HTML encoding using a default HTML decoder. The method actually is an equivalent of DecodeFromHtml and added just as an antipode for EncodeForHtmlAttribute method of the Encoder subsystem API.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the HTML category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromXml(): string | Converts the string instance from XML encoding using a default XML decoder.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the XML category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromXmlAttribute(): string | Converts the string instance from XML encoding using a default XML decoder. The method actually is an equivalent of DecodeFromXml and added just as an antipode for EncodeForXmlAttribute method of the Encoder subsystem API.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the XML category.  Throws the NullInputDataException when empty string object was used. |
| DecodeStringFromBase64(): string | Converts the string instance from BASE 64 encoding using a default XML decoder. Than the decoded bytes sequence is treated as a UTF8 encoded text.  Returns a converted string.  Throws a DecoderInputDataException exception when the initial string cannot be decoded from BASE 64.  Throws the SubsystemInputParamException when no default engine was specified for the BASE 64 category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromCss(): string | Converts the string instance from CSS encoding using a default CSS decoder.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the CSS category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromGZip(): byte[] | Decompresses the input string treating it is a BASE 64 encoded compressed binary data using a default GZIP decoder.  Returns decompressed bytes sequence.  Throws a DecoderInputDataException exception when the initial string cannot be decoded from BASE 64 or when a compressed data is corrupted (or is not a compressed data at all).  Throws the SubsystemInputParamException when no default engine was specified for the GZip category.  Throws the NullInputDataException when empty string object was used. |
| DecodeFromLdap(): string | Converts the string instance from LDAP encoding using a default LDAP encoder.  Returns a converted string.  Throws the SubsystemInputParamException when no default engine was specified for the LDAPcategory.  Throws the NullInputDataException when empty string object was used. |

## General API

In some cases a simplified API cannot be invoked due to some limitations or is inconvenient for some reason. There is a general API to use the Decoder subsystem in such cases.

Decoder subsystem is accessible through the Decoder property of the MetraTech.SecurityFramework.SecurityKernel class.

The following methods are defined in the API:

| Method | Description |
| --- | --- |
| Execute(engineId: string, input: ApiInput): ApiOutput | Invokes a decoder engine with the specified ID.  Returns a decoding result.  Throws the SubsystemInputParamException when an engine with the specified ID not found.  Throws the NullInputDataException when NULL or empty string passed to the decoder as an Input. |
| ExecuteDefaultByCategory(categoryName: string, input: ApiInput): ApiOutput | Invokes a default decoder of the specified category.  Returns a decoding result.  Throws the SubsystemInputParamException when a category with the specified name not found or a default engine was not specified for the category.  Throws the NullInputDataException when NULL or empty string passed to the decoder as an Input. |
| GetEngine(engineId: string): IEngine | Retrieves an engine the specified ID.  Throws the SubsystemInputParamException when an engine with the specified ID not found. |
| GetDefaultEngine(categoryName: string): IEngine | Retrieves a default engine for a category with the specified name.  Returns null when no default engine specified for the category.  Throws the SubsystemInputParamException when a category with the specified name not found. |
| GetEnginesForCategory(categoryName: string): IEngine[] | Retrieves all engines registered for the category with the specified name. |

## API Examples

### Simplified API

usingMetraTech.SecurityFramework;

…

stringinput = "&lt;br/&gt;";

stringactual = input.Decode(DecoderEngineCategory.Html + ".V1"); // returns "<br/>"

…

stringinput = "%20str%20";

stringactual = input.DecodeFromUrl(); // returns " str "

…

stringinput = "&lt;br/&gt;";

stringactual = input.DecodeFromHtml(); // returns "<br/>"

…

stringinput = "&lt;hrsize=&#34;2&#34/&gt;";

stringactual = input.DecodeFromHtmlAttribute(); // returns "<hrsize=\"2\"/>"

…

stringinput = "&lt;hrsize=&#34;2&#x22;/&gt;";

stringactual = input.DecodeFromXml(); // returns "<hrsize=\"2\"/>"

…

stringinput = "&lt;hrsize=&#34;2&#x22;/&gt;";

stringactual = input.DecodeFromXmlAttribute(); // returns "<hrsize=\"2\"/>"

…

stringinput = @"\3c span\20 style\03d \0022 font\0002d family\00003a\20 LU beck\22";

stringactual = input.DecodeFromCss(); // returns "<spanstyle=\"font-family: LU beck\">"

…

stringinput = "H4sIAAAAAAAEAOy9B2AcSZYlJi9tynt/SvVK1+B0oQiAYBMk2JBAEOzBiM3mkuwdaUcjKasqgcplVmVdZhZAzO2dvPfee++999577733ujudTif33/8/XGZkAWz2zkrayZ4hgKrIHz9+fB8/Ij7/qWKVzvJpNcvrdLFu2nS9nFaLVZ03TVosV+s2bfJftM6X0zydXKefv/gq/UGxStdtURbt9fj/CQAA///aod17PwAAAA==";

byte[] actual = input.DecodeFromGZip(); // returns array of bytes

…

stringinput = "abcd\\2A\\28\\29\\2F\\5c111";

stringactual = input.DecodeFromLdap(); // returns "abcd\*()/\\111"

### General API

usingMetraTech.SecurityFramework;

…

stringinput = "&lt;br/&gt;";

stringactual = SecurityKernel.Decoder.Api.Execute("Html.V1", input); // returns "<br/>"

…

stringinput = "%20str%20";

stringactual =SecurityKernel.Decoder.Api.ExecuteDefaultByCategory(DecoderEngineCategory.Url.ToString(),input); // returns " str "