MetraTech Security Framework

1. Introduction

This document provides a set of high level functional requirements for the MetraTech Security Framework. It defines key framework features and their intended functionality.

The interface specifications and the detailed functional specifications will be provided in the future design documents.

2. Overview

MetraTech product security has become an important business objective once the products gained a significant visibility. A number of efforts have been made to improve the security of the MetraTech products; however, a lot of product and process enhancements need to be made to achieve the necessary level of product security.

Some of the product security problems include the following:

- Limited ability to handle malicious environments (as the MetraTech products gain greater visibility and deployment exposure they'll become attractive targets for attackers subjecting them to great numbers of attack varying in levels of sophistication).
- Insufficient input validation and output encoding capabilities (regular expression based input validation provides only basic levels of protection; one type of output encoding is not sufficient because different output targets have different encoding requirements).
- Missing advanced security functionality (it's important to detect and react to unsuccessful attacks because they are pre-cursers to the future attacks that might be successful).

The main goals for the MetraTech Security Framework is to provide a unified framework for all security related functionality and to provide the necessary tools to protect the MetraTech product deployments from all types of attacks.

2.1 Requirements Outline

The MetraTech Security Framework must address the following key secure software development tasks:

- Input and output data type and content verification (to validate that the input and output data matches the expected data types and data value ranges/formats).
- Suspicious and malicious data detection (to find out if the validated input and output data contains unsafe values or to find out the reasons why data failed input or output validation).
- Input and output data encoding (to ensure that the data is safe for processing by various target MetraTech components and web browsers).

- Input data normalization and sanitization (to ensure that all MetraTech software components operate on data in the same format).
- Enforcing access control to various file system and application components (to ensure that user operations do not extend beyond the expected/allowed set of data object operations).
- Hiding internal data object information (to ensure that malicious users can't guess the internal data object information they should not access).
- Password value verification (to ensure the user select passwords are safe and to detect password guessing attacks).
- Data encryption and hashing using various encryption and hashing algorithms depending on the nature of the protected data (to ensure that the data is protected using the techniques best suited for their type and use; to ensure that the data protection is compatible with the protection algorithm upgrade requirements and with the protected data migration requirements).
- Application level intrusion detection (to detect attacks that can't be found through individual security checks when input/output validation or other checks are performed; to provide historical context to ensure more accurate malicious activity detection).

2.2 Implementation Constraints

MetraTech uses several programming languages (C#, C++, and VisualBasic), which requires the Security Framework to be accessible from all of those languages.

To simplify the framework development process and to eliminate the need for redundant code the MetraTech Security Framework should be implemented in C# exposing .NET assemblies as COM objects, so the code can be used in the Classic ASP VisualBasic and C++ components.

It should be noted that a small part of the C++ APIs will require native (non-COM implementation) for performance reasons. These performance critical framework APIs will be formally identified in the future design documents.

2.3 Dependencies

The MetraTech Security Framework is a base application component and because of that it must not have dependencies on general MetraTech code components. It's meant to be a stand-alone project that would be maintained and built independently from the Core MetraTech code base to ensure its integrity. The Security Framework will be integrated with the Core code base as a set of libraries.

The MetraTech Security Framework might have dependencies on third party software components. It will use some parts of the Microsoft Web Protection Library (a successor to the AntiXSS library).

2.4 Usage Guidelines

The MetraTech Security Framework follows the explicit usage model, which means that the core product code will use all, some, or no Security Framework features. This explicit usage model makes it possible to balance the performance and security needs of the product components when input or output data is processed.

In general, when the "Subsystem" Security Framework APIs are used (e.g., input validation subsystem) the product performance profile will change because additional data processing steps would need to be performed. However, when the "Utility" Security Framework APIs are used (e.g., hashing library) the application performance should not change significantly because no additional processing steps are introduced.

Most of the Security Framework capabilities will be heavily used by the web application components that have to deal with various external data going into the MetraTech products and the internal data leaving the MetraTech product components.

Some of the access control, encryption, and password handling functionality will also be used internally by the MetraTech product components.

It should be noted that some internal components such as MetraFlow will have very little interaction with the Security Framework subsystems because the data and activity analysis, validation, and filtering should be performed by the outer layers in the MetraTech product.

3. Framework Features

The high level security framework requirements outlined in the previous section represent the key application security problem domains. These application security problem domains are addressed by the following top level Security Framework features:

- Validator;
- Detector;
- Encoder;
- Normalizer;
- Encryptor;
- AccessController;
- ObjectReferenceMapper;
- RandomObjectGenerator;
- PasswordQualifier;
- SecurityMonitor;
- Processor.

Some of these features will be represented by libraries with public API while others will be stand-alone subsystems performing asynchronous processing and data analysis across multiple operations, sessions, and users.

The library-based framework features will need to be usable as stand-alone components as well as building blocks in more complex MetraTech Security Framework features.

The subsystem-based framework features will also need to have one or more sets of APIs that the MetraTech product components can use to interface with these security framework subsystems.

3.1 Validator

Validators are responsible for confirming one or more data properties such as data object types and data object value ranges or more complex properties such as the structure and the organization of the data content.

Validators can be grouped into the following general categories:

- Basic data type validators – to verify that a data object conforms to the basic properties of a basic data types such as integers, doubles, dates, and strings.
- Partial data object validators – to examine one of many data object properties (e.g., to verify that an integer parameter value fits within the range constraints of the examined parameter where the parameter range constraint can be significantly strictly than the range supported by the underlying integer basic data type).
- Application/complex data object validators – a set of data object checks that may also use partial data object validators to confirm that the data object value fits the specific requirements of a given application data object.

All validators must follow the validator filter API where all validator filters can be accessed through the main MetraTech Security Framework Validator API.

3.2 Detector

Detectors are responsible for discovering high level data object value properties.

Detectors can be grouped into the following general categories:

- Encoding detectors – to discover the types of encoding applied to the examined data.
- JavaScript detectors – to discover if all or some of the data can be considered JavaScript.
- SQL detectors – to discover if all or some of the data can be considered SQL commands.
- Shellcode detectors – to discover if all or some of the data can be shellcode used in exploits.
- OS command detectors – to discover if all or some of the data can be operating system shell commands.
- OS shell output detectors – to discover if all or some of the data can be operating system shell output

All detectors must follow the detector filter API where all detector filters can be accessed through the main MetraTech Security Framework Detector API.

3.3 Encoder

Encoders are responsible for encoding data that might contain parts with formatting incompatible or unsafe for various types of data processing.

Encoders must be able to encode data to be used safely in:

- CSS.
- HTML attributes.
- HTML
- JavaScripts.
- URLs.
- XML attributes.

- XML.

All encoders must follow the encoder filter API where all encoder filters can be accessed through the main MetraTech Security Framework Encoder API.

3.4 Normalizer

Normalizers are responsible for converting various data representations into one standard format that can be used for data processing. The normalization process might involve the use of decoders.

Normalizers can be grouped into the following general categories:

- Decoders – to remove one or more data encodings applied to the data being processed.
- Senitizers – to extract data the application wants to know and to discard the rest.
- Equalizers – to convert from multiple equally valid representations into one selected format.

All normalizers must follow the normalizer filter API where all normalizer filters can be accessed through the main MetraTech Security Framework Normalizer API.

3.5 Encryptor

Encryptors are responsible for low level functionality such as encryption and hashing algorithms as well as high level functionality such as data object type specific protection.

Encryptors must provide support for the following data categories:

- Passwords.
- Credit card numbers.
- Web application parameters (in addition to being encrypted the parameter data needs to be constrained by the parameter type, timestamp, and session information to prevent replay attacks).

Some of the encryption and hashing algorithm implementations might by provided by the OS while others will require full implementation (e.g., bcrypt hashing).

3.6 AccessController

AccessController is responsible for enforcing the access requirements to the following types of objects:

- File objects.
- Application variable objects.
- Application functions.
- URLs.

3.7 ObjectReferenceMapper

ObjectReferenceMapper is responsible for providing a proxy for internal data objects to allow the application to control what application objects are exposed and to what sessions. This makes it possible to prevent forceful browsing and direct object manipulation attacks that can be successful when internal data object information is exposed directly to the application users.

3.8 RandomObjectGenerator

RandomNumberGenerator is responsible for generating the following types of random data:

- Numbers.
- GUIDs.
- Passwords.

3.9 Processor

Processors are responsible for combining multiple processing steps into processing chains. Processing steps can include different types of the MetraTech Security Framework features and other processors making complex composite processing possible.

Processors can be grouped into one of the following categories:

- Request Processors that look at application request properties, URL, and query arguments.
- Input Object Processors that look at individual query parameters or header fields.
- Output Object Processors that look at individual result data objects.

All processors must follow the processor filter API where all processor filters can be accessed through the main MetraTech Security Framework Processor API. The main MetraTech Security Framework Processor API must also provide the necessary infrastructure to store and lookup processors to allow MTSF users to offload the processor management and data binding operations.

3.10 PasswordQualifier

PasswordQualifier is responsible to ensuring the password quality. In addition to providing the security industry standard regex based password qualifications it must provide additional password complexity qualifications that are impossible to express using regular expressions (character proximity checks, dictionary checks, fuzzy dictionary checks, etc).

3.11 SecurityMonitor

SecurityMonitor is responsible for providing the infrastructure for collecting and processing security related application events to detect suspicious and malicious activities that can’t be detected when individual data objects or application operations are analyzed.

MetraTech products must explicitly enable the SecurityMonitor inspection points to take advantage of the SecurityMonitor capabilities. The SecurityMonitor capabilities are configured using the SecurityMonitor policies that control inspection point parameters, thresholds, and response actions.

The SecurityMonitor policies must support the following actions:

- Block operation – to block an individual user operation.
- Block user – to block a specific user.
- Block address – to block a specific IP address.
- Log – to log an application security event.
- Redirect – to redirect an operation to an alternative location.

- Transfer - to transparently transfer the current operation processing to an alternative processing engine.
- Logout user – to logout the current session user.
- User security warning – to send back a visual security warning to the current session user.
- Application administrator notification – to send an admin notification.
- Session parameter change – to change one of the current session parameters (e.g., change caching settings, change timeout settings, change security thresholds, etc).

The SecurityMonitor inspection points can be grouped into the following categories:

- Request Monitors (Unexpected HTTP Commands; Attempts to Invoke Unsupported HTTP Methods; GET When Expecting POST; POST When Expecting GET).
- Authentication Monitors (Use of multiple usernames; Multiple failed passwords; High rate of login attempts; Unexpected quantity of characters in username; Unexpected quantity of characters in password; Unexpected types of characters in username; Unexpected types of characters in password; Providing only the username; Providing only the password; Adding additional POST variables; Removing POST variables).
- Access Control Monitors (Modifying URL arguments within a GET for Direct Object Access attempts; Modifying parameters within a POST for Direct Object Access attempts; Force browsing attempts; Evading presentation access control through custom posts).
- Session Monitors (Modifying existing cookies; Adding new cookies; Deleting existing cookies; Substituting another user's valid session ID or cookie; Source IP address changes during session).
- Input Data Processing Monitors (Cross Site Scripting Attempt; Violations of implemented White Lists; Double encoded characters; Unexpected encoding used; Blacklist inspection for common SQL injection values; NULL byte character in file request; Carriage Return or Line Feed character in file request).
- Output Data Processing Monitors (Detect abnormal quantity of returned records; Detect system shell command result information).
- File I/O Monitors (Detect large individual files; Detect large number of file uploads).
- User Trend Monitors (Irregular use of application; Speed of application use; Frequency of site use; Frequency of feature use).
- Application Trend Monitors (High number of logouts across the site; High number of logins across the site; High number of same transaction across the site).