

**HEAT** Next Generation

Common Interface

###### 

###### 

|  |  |
| --- | --- |
| **Date:** | 7th Apr 2008 |
| **Customer:** | Holodeck Next Generation |
| **Author:** | Rahul Chaturvedi (rchaturvedi@securityinnovation.com) |

Security Innovation, Inc.

187 Ballardvale Street, Suite A170

Wilmington, MA 01887

+1.978.694.1008  
  
www.securityinnovation.com

# Background

HEAT NG will provide a common interface to specify interceptions across all domains. This interface is essentially a C++ API that allows a user to flexibly provide data on what kind of function they want to intercept. This interface needs to be abstract enough to handle future domains and interception types.

# Basic Design

HEAT needs the following information from the user to set up an intercept;

* Domain: Currently these will be only 'native' and '.net'; a 'java' domain is on the drawing board
* Original function identification: This can be specified in multiple ways depending on the domain
* Imposter function identification: This can also be specified in multiple ways depending on the domain
* Object Instance filter: To enable object based interception, we need to also specify a filter which will specify interception only of particular objects

These are the factors that can be used to provide function identification information to the interception engine, ordered by domain;

Win32:

* Virtual Address
* Module Name + Offset
* Module Name + Function Name
* Module Name + Class Name + Method Name

.NET:

* Assembly Name + Function Name
* Assembly Name + Namespace + Function Name
* Assembly Name + Namespace + Class Name + Member Name

# Design Details

## API

The base API will be provided by the client and will have only one method;

* intercept

This method will be overloaded to either accept an intercept object or a list of intercept objects.

It will have another set of overloads to accept object filter instances as an additional parameter.

## Interception Objects

The interception objects will inherit from the base intercept class. This class will provide the following pure virtual methods that will need to be overridden by any subclasses.

* getDomain: This method will return a string identifying which domain this interception object belongs to
* setHEATHook: This method will accept a function pointer, which will point to the HEAT's notification function. This method will be used to associate an interception object with a particular HeatNG Client

The intercept base class will provide an additional two methods, which may be overridden by future intercepts subclasses but will not be currently.

* enableIntercepts: This will be used to enable a disabled intercept instance. All instances will be created in an enabled state, so once they are sent to a client, they are instantly set to be intercepted
* disableIntercepts: This function will disable an enabled intercept

These methods will use the heatNG hook, setup during intercept, to communicate with the client that the state of this intercept has been updated. The client will then communicate with the server running in the AUT to perform the enable or disable.

To ensure error handling, neither of these functions will return till the AUT has confirmed action on the enable or disable command.

## Object Filter Objects

An object filter object is an instance of the ObjectFilter class. This class provides the functionality to specify filtering options to enable object instance oriented interception.

The design for this class is provided in the Object Based Interception design document.

## Implementation notes

The intercept objects will be mirrored in the heatNG server they are meant for. If two intercept objects map to the same intercept, or if an intercept is not found, all of these errors will be detected in the server and passed on in real-time to the intercept call in the client.

Once intercepted, the user does not have the option to 'unintercept' a function. This functionality is essentially the same calling the disableIntercept method on the object, and is considerably more complex to implement.