

**HEAT** Next Generation

Architecture

###### 

###### 

|  |  |
| --- | --- |
| **Date:** | 3rd Mar 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 is being written from the ground up to address several key design issues with the current version. The goals behind the design are essentially to provide,

* Extensibility
* Abstraction
* Performance

A complete list of features required of HEAT NG are listed in the "HEAT NG - Features" document.

# Basic Design

These are the primary components of HEAT NG. Keep in mind that HEAT NG is a framework that allows interception and provides no interception mechanism of it's own save what is required to setup the plug-ins for various interception domains that have been written for it. The only intercepts that the HEAT NG will set up will be certain 'insert' points that [only] a domain interception plug-in will be able to use.

These will be the functional areas provided by HEAT NG's base framework

1. Client: Any process that uses our interception API to specify intercepts for an application
2. Server: A server component that will inject into a target process and facilitate the loading of domain interception plug-ins
3. Agent: An agent that will facilitate higher privilege operations
4. Communications Layer: A common communication framework to be used across domain interception plug-ins and API code
5. Plug-in mechanism: This mechanism will allow us to write each individual domains interception separately and the HEAT NG framework will load it as and when required
6. Native Interception Layer: This will be a layer that will provide interception for native functions. This layer will be simplistic, working only with Virtual Addresses for both the original and intercepted functions. This layer will be the only section of HEAT NG which will be platform specific

The HEAT NG basic framework will have as minimal platform specific code as possible, most of it will be coded in C++. This is what a final HEAT NG framework in operation would look like;

HEAT NG Agent

## A.U.T.

Comms Layer

Plug-ins

HEAT NG Client

(any process that uses the API)

.NET Interception

Native Interception

# Design Details

## HEAT NG Client

The client is typically going to be the user application (in our case, Holodeck NG). These are its functions;

* Create an AUT object with all the information about the application
* Before we launch the AUT (or attach to it), we 'need' to have all the interception information in hand
  + Once the AUT launches or we attach, we will not allow more functions to be intercepted
  + The user has the option to intercept more functions than he wants and have the interception disabled; but he does need to specify which functions he will need intercepted during the entire lifecycle of the AUT process, in advance
* On the cue from the user, either launch the HEAT NG Server manually, or launch it through the agent (depending on the target process type)
* The API can now be used to enable/disable existing intercepts, enable/disable logging, etc

## HEAT NG Server

The HEAT NG Server will essentially be a thread started by the first DLL we inject into a process. This server's essential tasks are,

* Setup the communication interface: Only one client at a time will be able to connect to this interface, but, we will have extra connections allowed for diagnostic calls or calls from the agent
* Start the requisite plugins: The server will read through the list of intercepts, tabulating which domain plug-ins does it need to start. The ones that are available then will be started and the initial intercept information passed to them
* After this the thread will primarily sit idle, waiting for any more commands (particularly the quit command)
  + A decision needs to be made here eventually; do we want this thread around even anymore?

## Agent

The only role the agent is going to have as of now, is to execute clients commands in its own process space. These can include either starting up the HEAT NG Server or communicating with it

## Communications Layer

The comms layer will be Winsock based. It will be used for communication between all parts of HEAT. It needs to provide these services;

* Ability to start a comms server for any component requesting it
* Ability to connect to a comms server for any component
* Provide unique identification for each comms interface created (which will translate through a Winsock port through some sort of hashing)
* A clear command protocol. This protocol will either handle,
  + Commands: Instructions to execute a specific command
  + Data: Regular binary data sent between components

## Plug-in mechanism

This mechanism will allow the HEAT NG Server to launch various plug-ins which will provide domain specific interception. At the time of design, we're expecting to have plug-ins at least for Native Interception, .NET Interception and Java Interception.

* A HEAT NG plug-in will provide a module which will contain its code.
  + The module will need to have an entry point for setup. This entry point will be called by the HEAT NG Server
  + The module will need to have an entry points for insertion point intercepts
    - What this means is that the module will need to have exposed intercept functions for certain Operating System API's. These intercept functions, at the least, must forward the call to the real Operating System API that it is intercepting (a point to which will be provided)
    - Currently, the OS API's that a plug-in can provide intercepts functions for are CreateFileW and LoadLibraryExW (on Windows). If intercept functions aren't provided for these calls, the calls will be untouched
  + The module needs to start up a communications server at which it will accept certain commands to enable/disable intercepts and to provide diagnostic information
* So once a plug-in is loaded, it will,
  + Setup its intercepts table
  + Start the communications server
  + Intercept the various calls it needs to (this may be initially a bit delayed, depending on when the insertion point function gets called)
* At this time, attaching to a running process for any plug-ins other than native is not supported. Other than native, it probably may never be supported due to several real world issues with attaching to running processes. If a business case to do so shows up, we will work on it then

## Native Interception Layer

The native interception layer is described in the " HEAT NG - Plug-in - Native Interception" document.