ThreatFactor NSIA - Bug #197

Issues with ThreatScript saving of arrays

09/25/2010 02:01 AM - Luke Murphey

Status:	Closed	**Start date:**	09/25/2010
Priority:	Normal	**Due date:**	
Assignee:	Luke Murphey	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0 (Release)		

Description

Current issues are:

1. System loads values multiple times, it may not be deleting old items correctly
2. System cannot load some values, saying it is not serializable though it implements the Serializable interface

History

#1 - 09/25/2010 01:13 PM - Luke Murphey

The root cause has been identified:

The ScriptDefinition class is serializing objects as arrays containing NativeJavaObject's. This must be unwrapped to the original object in order to be serialized correctly. The fix is to have the environment class convert the NativeArray object to an Object array and unwrap each NativeJavaObject contained within.

#2 - 09/25/2010 01:16 PM - Luke Murphey

- Status changed from New to In Progress

- Assignee set to Luke Murphey

- Target version set to 1.0.1

#3 - 09/25/2010 02:16 PM - Luke Murphey

One problem that has become apparent is that the ScriptDefinitions deserialize object arrays as basic arrays that do not include the push and pop functions that the JavaScript arrays include. Therefore, the array that is deserialized must be converted into a JavaScript array manually in order to call push and pop.

#4 - 09/25/2010 02:24 PM - Luke Murphey

The .SuspiciousContent.Cross_Domain_Scripting was modified to support the saving of script domains. Below is an updated script that is confirmed as working with the new script engine changes. Note that the baseline function converts the array into a native Javascript array in order to call push and pop.

```
/*
 * Name: Audit.SuspiciousContent.Cross_Domain_Scripting2
 * Version: 4
 * ID: 1000216
 * Message: Reference to a client-side script residing on a separate domain
 * Severity: Low
 * Reference: url, ha.ckers.org/xss.html
 * Reference: url, en.wikipedia.org/wiki/Cross-site_scripting
 */

importPackage(Packages.ThreatScript);
importPackage(Packages.HTTP);

function stripSubDomain( domain ){
    var re = /([0-9A-Z_.]+\.)?([0-9A-Z_]+\.[0-9A-Z_]+)/i;
    matches = re.exec(domain);
```

```
    var result = null;
    if( matches == null ){
        result = domain.toLowerCase();
    }
    else{
        result = matches[matches.length - 1].toLowerCase();
    }

    //Make sure the result is not an empty string
    if( StringUtils.trim(result).length() == 0){
        return null;
    }

    return result;
}

function isInList( domain, scriptdomains ){

    if( scriptdomains == null ){
        return false;
    }

    for( i in scriptdomains ){
        if( scriptdomains[i] == domain){
            return true;
        }
    }

    return false;
}

function analyze( httpResponse, variables, environment ){

    //Get the existing scripts that have been accepted
    var acceptedScripts = environment.get("ApprovedScriptDomains");

    if( acceptedScripts != null ){
        acceptedScripts = acceptedScripts.getValue();
    }

    var parser = httpResponse.getDocumentParser();
    var location = new URL( httpResponse.getLocation() );
    var localhost = stripSubDomain( location.getHost() );
    var domainScripts = [];

    //Get a list of all script tags
    var tagNameFilter = new TagNameFilter("script");
    var nodesList = parser.extractAllNodesThatMatch(tagNameFilter);

    //Analyze each script tag to determine if any have a non-local script reference
    for( var c = 0; c < nodesList.size(); c++ ){

        var tag = nodesList.elementAt(c);
        var src = tag.getAttribute("src");

        if (src != null){
            try{
                var domainScript = new URL( location, src );

                domainScript = stripSubDomain( domainScript.getHost() );

                //Determine if the URL is absolute and ensure that the location is local
                if( domainScript != null && domainScript != localhost ){

                    if( isInList(domainScript, acceptedScripts) == false
 && isInList(domainScript, domainScripts) == false ){
                        domainScripts.push( domainScript );
                    }
                }
            }
            catch(e){
                /*
                 * Ignore this exception, this can occur when the HTML is malformed. A malformed script
                 * source tag shouldn't be followable and therefore should not be a threat.
                 */
```

```
            }
        }
    }

    // Alert that a cross-domain script was discovered
    Debug.sendMessage( "domainScripts.length=" + domainScripts.length );
    if( domainScripts.length > 0 ){
        var desc = null;
        var changed = false;

        // Get the list of scripting domains that have been discovered already
        var discoveredScripts = environment.get("DiscoveredScriptDomains");

        if( discoveredScripts != null ){
            discoveredScripts = discoveredScripts.getValue();
        }

        if( discoveredScripts == null ){
            discoveredScripts = [];
        }

        for( i in domainScripts ){
            if ( desc == null ){
                desc = domainScripts[i];
            }
            else{
                desc = desc + ", " + domainScripts[i];
            }

            //Add the script to the discovered scripts list
            if( isInList( domainScripts[i], discoveredScripts ) == false ){
                discoveredScripts.push(domainScripts[i]);
                changed = true;
            }
        }

        // Save the list of discovered script domains (if changed)
        if( changed && discoveredScripts != null ){
            environment.set("DiscoveredScriptDomains", discoveredScripts, false);
        }

        return new Result( true, "Cross domain scripts discovered: " + desc );
    }

    return new Result( false, "No cross domain scripts detected" );
}

function baseline( environment ){

    // Get the list of scripting domains that have been discovered already
    var discoveredScripts = environment.get("DiscoveredScriptDomains");

    if( discoveredScripts == null ){
        return true;
    }
    else{
        discoveredScripts = discoveredScripts.getValue();
    }

    if( discoveredScripts == null ){
        return true;
    }

    // Get the list of domains that have been aproved for scripting
    var approvedScripts_old = environment.get("ApprovedScriptDomains");
    var approvedScripts = [];

    if( approvedScripts_old != null ){
        approvedScripts_old = approvedScripts_old.getValue();
    }

    if( approvedScripts_old != null ){
        for( i in approvedScripts_old){
            approvedScripts.push(approvedScripts_old[i]);
        }
```

```
    }

    // Update the list of approved scripting domains
    for( i in discoveredScripts ){
        if( isInList(discoveredScripts[i], approvedScripts) == false ){
            approvedScripts.push(discoveredScripts[i]);
        }
    }

    // Save the approved scripting domains
    environment.set("ApprovedScriptDomains", approvedScripts, false);

    // Clear the discovered script domains
    environment.remove("DiscoveredScriptDomains");

    return true;
}
```

#5 - 09/25/2010 05:44 PM - Luke Murphey

- Status changed from In Progress to Closed

- Target version changed from 1.0.1 to 1.0 (Release)

- % Done changed from 0 to 100