NASK GOV<>CERT.NL



2.0

Specification of HSN 2.0 JavaScript Static Analyzer

Paweł Jacewicz

Version 0.3
Relevant issues: #4925

Last edit by: Łukasz Siewierski, 2012-11-08
Sprint: 11

Summary
This document specifies operation and configuration of the JavaScript Static Analyzer (`js_sta`).

Version	Date	Author	Changes
0.1	2011-11-23	Paweł Jacewicz	Initial version
0.2	2011-11-24	Paweł Pawliński	Many fixes throughout the document: more specific descriptions, improved local parameters
0.3	2012-11-08	Łukasz Siewierski	Added information about context whitelisting

Contents

A. JS Static analyzer service description 3
B. JavaScript Static Analyzer processing 4
C. Local service configuration 5

A. JS Static analyzer service description

The service should be implemented on the basis of research on capabilities of HSN 1.5 low-interaction client honeypot [1] and a proof of concept implementation of the LIMv2 tool. Functionality in terms of classification of JavaScripts should not be lesser than in LIC [1] (taking into account only ngrams, naive Bayes classifier and keywords confirmation).

The service will receive an URL object containing a list of JavaScript contexts, retrieve the source code, process it individually and save results in attributes of the current object (see data contract [2] for detailed information). The service will not create new objects. There should be a possibility of running multiple classification threads in parallel. Number of threads will be constrained by the service configuration. Keyword lists will be provided as service parameters in a workflow definition.

Internally, the Weka Toolkit [3] is used for classification, training data for model creation is supplied separately. Training dataset is contained in a single ARFF file which is specified in the service configuration. Once the training data is read and a classifier model is created, this file should not be accessed any more (all the processing threads should share the model). There should be a simple way of reloading training data and creating a new model in runtime, without disrupting already running classification tasks.

Whitelisting is performed against a supplied file that contains hashes (one per line). If the context hash matches hash in the file, context is said to be "whitelisted". Hashing algorithm strips context only to alphanumeric signs (A-Z, a-z and 0-9) and then calculates MD5 hash sum of this character string.

B. JavaScript Static Analyzer processing

JavaScript Static Analyzer service provides functionality to analyze JavaScript source code without executing it. The analysis is performed on chunks (contexts) of JavaScript code extracted by a web client (see data contract for web client [2] for details). The method of analysis is similar to one implemented in HSN 1.5 [1] — the service detects malicious and obfuscated JavaScript code. For this purpose the Weka Toolkit [3], ngrams and pattern matching mechanisms are used. The processing path of JavaScript contexts is presented in figure 1.

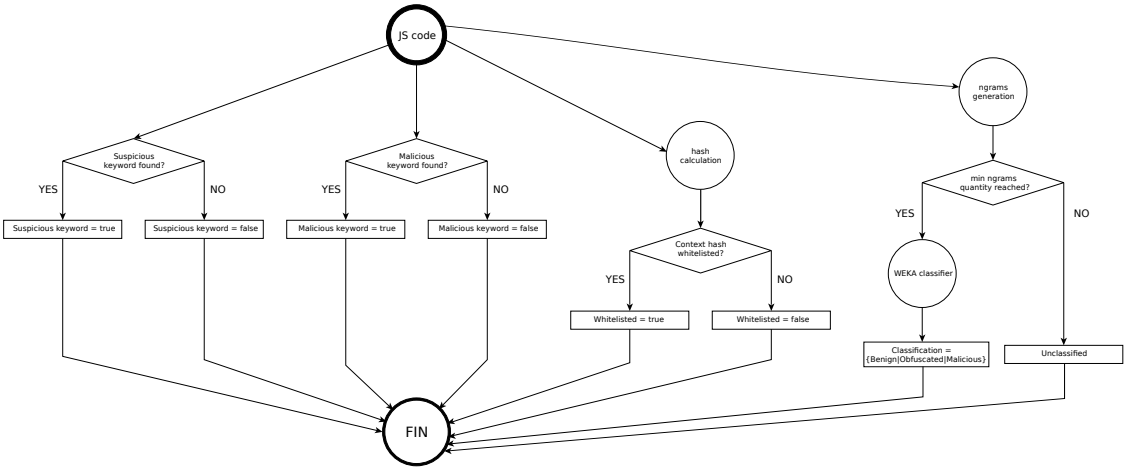


Figure 1.: Classification model

Processing steps are as follows:

1. Retrieve a JavaScript context together with its identifier (used to distinguish between different contexts in a single web page).
2. Perform in parallel:
   - check whether JavaScript contains any predefined malicious keywords
   - check whether JavaScript contains any predefined suspicious keywords
   - generate a list of most common ngrams
   - calculate context hash and compare it against the list of predefined hashes

3. Perform Weka analysis on generated ngrams.
   a) in case number of generated ngrams is insufficient skip the Weka analysis and assign classification *unclassified*
   b) assign classification according to Weka result: *malicious, obfuscated, benign*
4. Associate all detected keywords with the context they were found in.
5. Save all information gathered about all contexts (classification, keywords, whitelisting status) in a list of structured form in an attribute of the current object.
6. Save an overall classification for URL object based on results for all contexts. Assuming $unclassified < benign < obfuscated < malicious$ ordering of classifications, the overall result is the maximum of all context classifications.
7. Information whether any of the keywords on both lists were found during analysis and whether script is whitelisted or not should be added to the URL object. This can be done via relevant attributes containing boolean values.

C. Local service configuration

Local configuration is expressed through a set of parameters describing initial state of the running service. This document does not specify format of the configuration file.

`thread_number` mandatory: yes type: integer default value: 10
The number of classification threads the service is able to spawn at the same time. The number corresponds to maximum number of JavaScript code chunks the service is able to process simultaneously.

`training_set` mandatory: yes type: string default value: *platform-specific*
Path to the ARFF file that contains labelled training data that should be used by Weka for training a classifier.

`classifier_name` mandatory: yes type: string default value: `weka.classifiers.bayes.NaiveBayes`
Parameter declares the name of classifier to be used by Weka Toolkit.

`ngram_length` mandatory: yes type: integer default value: 4
Parameter declares the length of single ngram generated from the JavaScript source code. The length must be the same as used when generating the classifier model file.

`ngrams_quantity` mandatory: yes type: integer default value: 50
Number of most frequent ngrams that appear in a context (top n) that should be used in the classification process. It must be consistent with contents of the training dataset.

References

[1] HSN 1.5 Low-Interaction Component capabilities

[2] Data Contract Specification for HSN 2.0 Services

[3] Weka 3 Toolkit, `http://www.cs.waikato.ac.nz/ml/weka/`