OWASP ZAP: Dinamically Configurable Actions Add On

This document contains all the choices made by the developer during the developing of the project and describes how the developer arrived to these choices.

The main reference is the developer's blog: http://seccoalegsoc.blogspot.it/ (which contains the weekly reports and the proposal).

Community Bounding Period

In this phase the student is asked to hang out at the comminity of the organization.
In this phase he is also asked to think more in detail what to do of his proposal, how to start and the way in which to start the implementation of the project.

Using the submitted proposal as main reference, these are the ideas.

May 30th

The main goals of the proposal are divided into two important parts:

1. Improving of ZEST;
2. Improving/Creation of the ZAP ZEST add-on.

The work should start first with the ZEST scripts. The developing of complex conditionals is the first thing to make, in order to let user build more complex zest scripts in a simpler way.

The way to implement this is described here (last post by Simon Bennetts).

The usage of brackets is not necessary in the develop phase, but is a must-have for the graphical interface: User needs to create complex conditionals in the most familiar way.
The graphical UI (to decide if it is better to develop the complex-conditional UI at this point, I _think_ this could be a nice, and it is almost done: this one can be used with a port to the ZAP add-on).

After this phase, the loop described in the Timeline section of the submitted proposal starts.

The first loop will occur with this usecase:
Suppose that a WEB application detects an attack and logged the user out; the user will be able to detect this with a zest script and reauthenticate in a very easy way;

(this should be the simplest one).

The timeline of this usecase will be the following:

1. first the design the usecase (almost done);
2. develop hooks and integration;
3. creation of the documentation about the code already developed;
4. test.

June 4th

With the suggestion of the mentor Simon Bennetts, I started looking the code reguarding the ZAP Filters. They should be completely replaced by ZEST scripts.

June 5th

Looking at the code of the ZAP Filters in *org.parosproxy.paros.extension.filter* and looking at the *org.parosproxy.paros.extension* I was thinking about how to develop the Hooks for actions and transformation. The main goal is to replace the whole filter package with zest scripts, in order to guarantee much more flexibility.
A possible idea is to implement something like the following:

After a chat with the mentor we discussed about the following parts:

- Zest Scripts allowed inside every request/response.
- To create new usecases.

A first usecase could be the replacing of filters using Zest.

June 6th

Looked at the code inside *org.parosproxy.paros.extension.filter.*
Start developing the same utility with zest.
PHASE 1: DESIGN.

- Each filter is represented by only one Zest Script;
- One ZestRunnerThread per Filter;

June 7th

Continued developing Replace* filters as suggested by the mentor;

June 8th

Very first try to interact with the current code! Created first draft extension.

June 9th

Played with the code in order to replace Body values using Zest Script. I tried to change Responses using Zest Scripts. Unfortunally ZestTransformations works only with replacing of forms in the request.

June 12th

Created the repository. Configured Bodgeit in order to make some tests.
Stuck with the code:

1. Cannot interface correctly with ZAP!
2. Cannot understand the usage of ZestTransformFieldReplace.

June 13th

I tried to run some Zest script based on hand made Request.
Discussed with the mentor about the next steps of the work:

1. Design filter: very important as a feedback for Zest implementation;
2. Run Zest Script “inline”;

June 17th

Because of the preparation of a couple of exams, I stopped developing for a couple of days.
Design of ZestFilter:
https://docs.google.com/document/d/1KCdNCaJgE_09xrcGacOcXwVfQsupA49PBab78iMDnVY/edit .

June 19th

Continuing the design of ZestFilter (same link).
Wrote down Zest Pseudo Code for ZestReplaceFilter && created a draft of interface on how to implement Filter (recycling existing code).
Fixed some concepts with the mentor.

June 20th

Once fixed some concepts with the mentor yesterday, I proceeded with the the design of a set of classes which allow to run Zest Scripts in every extension of ZAP.
I try to recycle the work made for replacing ZestFilter to make it more generic!
Stuck on the refractoring of ZestFilter: maybe ZestEntry?

Note: all the name below need to be refractored. Don’t consider them as final!

Created Interfaces and Abstract Classes. I need to implement the complete implementation & to refractor the names. Now I have

1. ZestEntry, containing:
    a. boolean enabled;
    b. name
    c. ZestScript;
    d. ID;
    e. methods onHTTP[RequestSend,ResponseReceive].
2. ZestEntryCollector (a container of ZestEntries):
    a. List of ZestEntry;
    b. ExtensionZest (which will run the single ZestEntries);
    c. methods to create and load ZestEntries.

The first one is an interface which could describe a single ZestScript. The methods onHTTP are now abstract in the implementation.
I would like to provide 2 types of ZestEntry:

1. a standalone Script;
2. a Script which works in cascade with others.

The first one needs to have a reference to the ExtensionZest and needs to be independent by any other script (for example a script which tries many SQL Injection: user clicks on the single url and launch only that script);
The second one works in cascade with others (as in a filtering process). All the execution phase is demanded to the ZestEntryCollector.

June 21st

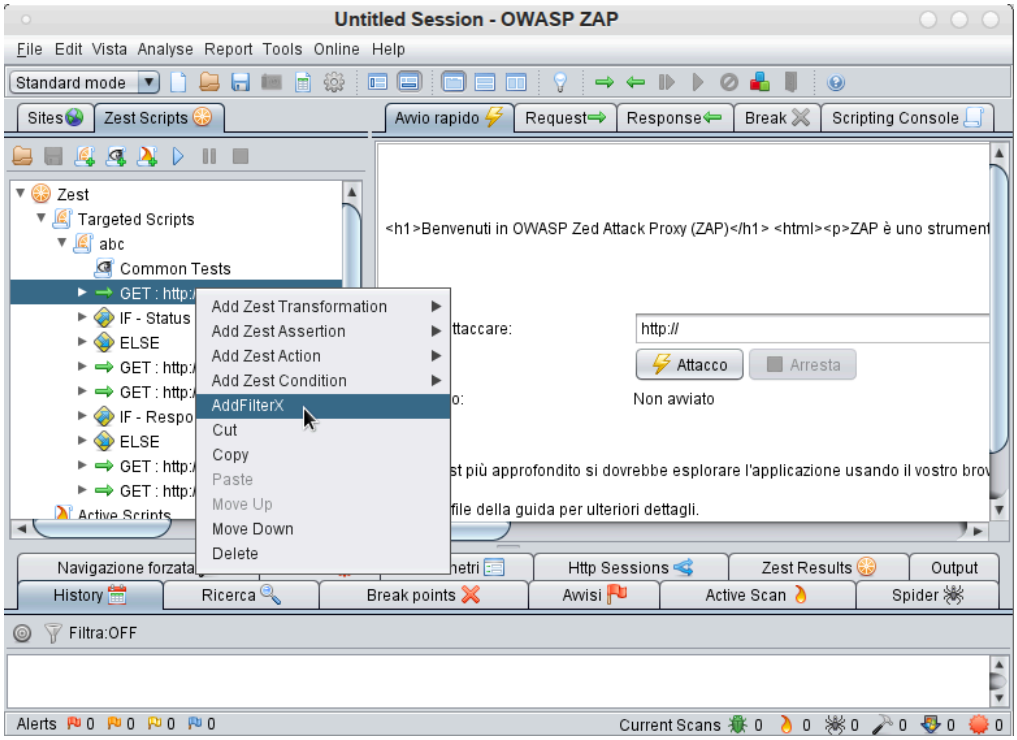
I tried to create a ZestSimpleScript object. This object should execute the ZestScript giving an URL and figures as a standalone Script. Then I tried to plug it into the ExtensionZest.
Had a chat with the mentor and fixed some targets.

June 22nd

First hands on code already checked in and continuing with Filter implementation.
Added classes ZestFilter & ZestFilterContainer.
Setting Swing Explorer [Done]
Corrected compiling problems in my repository.
Read some of the InternalDetails in Zap wiki;
Completed merging of the new code with mine.
Writing UI & UI logics.

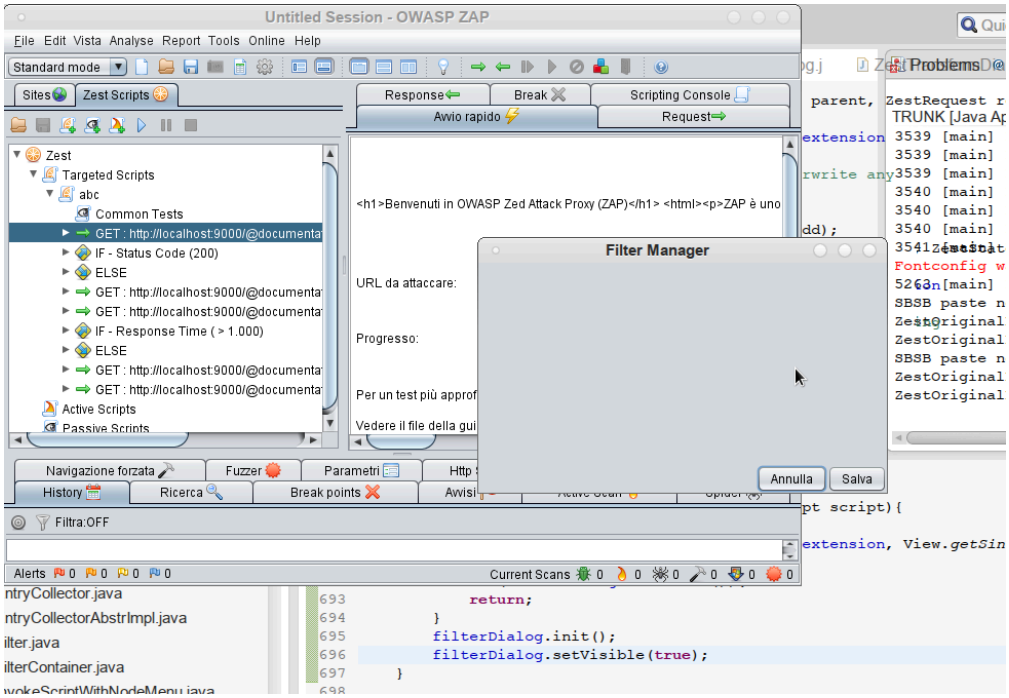
June 23rd

Added build folder to my own repository.
Added a voice in the UI menu to add a Script as Filter! (See next page)



I started the creation of the ZestFilterDialog
Stuck with the correct name to pass in the contructor for the class ZestFilterDialog.
Solved (The properties file).

I had to design the dialog for the Filter Managing and recall it from the Tools Menu.

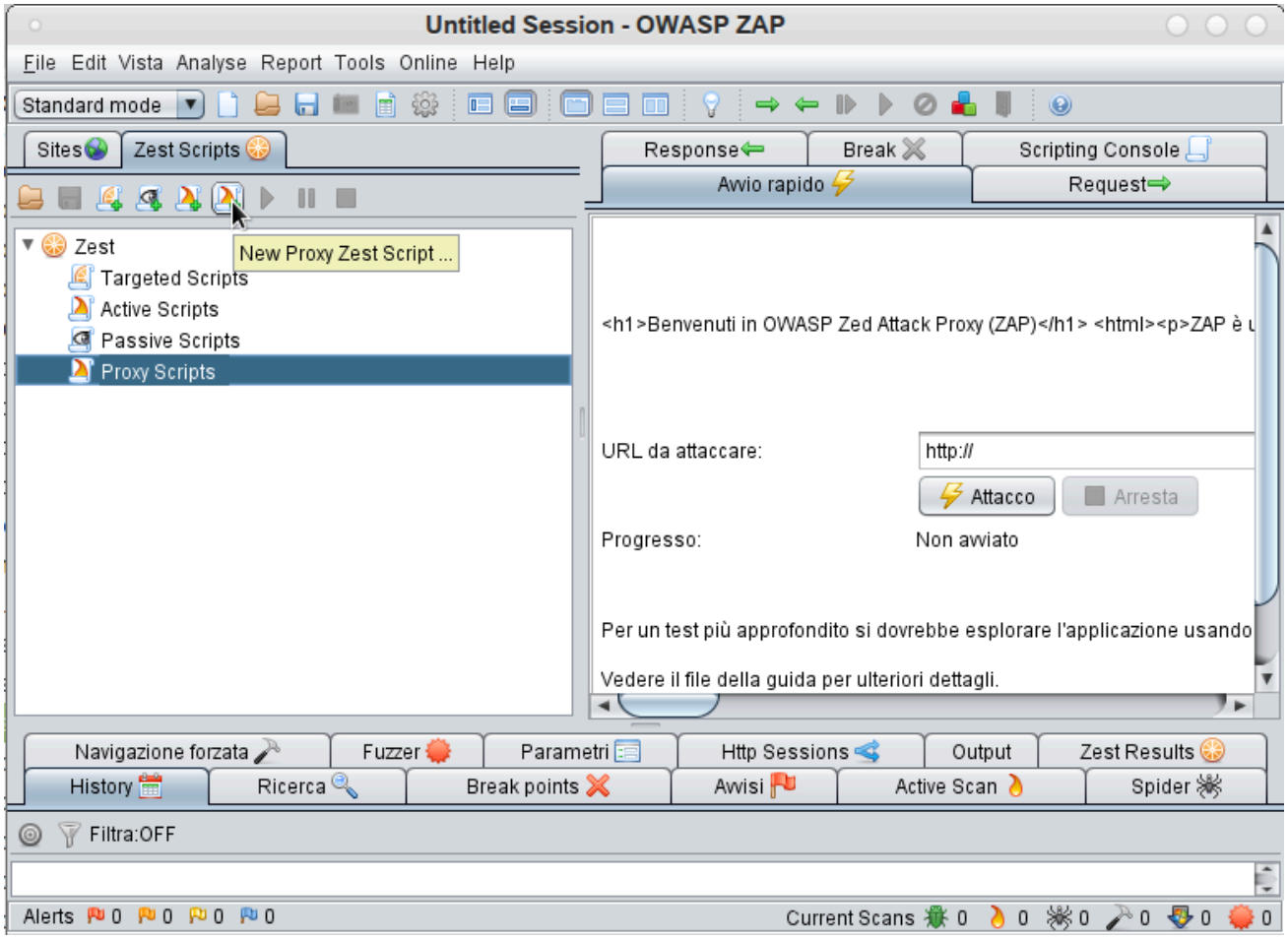


June 26th

Creating the top level node ProxyScript.
Added the top level nodes (Same picture as ActiveScript).
Needed changing in ZestScript class.

June 27th

Added ProxyScript top node in zest model:



To insert the right icon, just replace the current zest-proxyscript*.png files.
Added ZestScript.Type.Proxy
Exception raises:

java.lang.NoSuchFieldError: Proxy
at org.zaproxy.zap.extension.zest.ZestScriptsPanel$6.actionPerformed(Unknown Source)

I suppose mistook in building zest and in using it inside eclipse.

June 30th

I'm still trying to fix this strange problem.
The problem was that, even if the included jar contained the changes, the class files inside the zap extension didn't. Fixed roughly decompressing, changing the .class and recompressing the .zap file and everything seems to work, but I spent a bit of time...

July 1st

Looking again the usage of filters.

July 3rd

Implementing the ZestFilterDialog.
Fixing an exception: it seems to be related to a bug I found last days. Try to fix.
After a chat with the mentor, I'll pass to Proxy.

July 4th

Switching to Zest Complex Conditional.
refs:

- Google Group: https://groups.google.com/forum/#!topic/mozilla-zest/5Uhs67ogLkg
- My Repo: https://github.com/Vankar/zest

Considering brackes as implicit.
This is the Structure:

- ZestConditional
    - ZestBooleanAnd
        - ZestBooleanOr
            - ZestExpressionRegex (regex exp1)
            - ZestExpressionRegex (regex exp2)
        - ZestBooleanStatusCode (200)
    - List <ZestStatement> ifStatements etc...

Going more inside, I designed the following internal structure for each class/interface:

- abstract class ZestExpression extends ZestElement implements ZestConditionalElement:
    - List<ZestConditionalElement> children;
    - boolean not;
    - ZestConditionalElement parent;
    - String name;
    - static int counter; // for the default name
    - abstract boolean evaluate;
- class ZestConditional extends ZestStatement implements ZestContainer, ZestConditionalElement:
    - List<ZestConditionalElement> children;
    - List<ZestStatement> ifStatement;

    - List<ZestStatement> elseStatement;
- class ZestBoolean[And,Or] extends ZestExpression implements ZestConditionalElement:
    - ZestConditionalElement parent;
    - List<ZestConditionalElement> children;
    - boolean evaluate();
- interface ZestConditionalElement extends ZestContainer
    - getIndex(); //return the index of the statement;
    - getChildren();// returns the children;
    - isLeaf();//true if it has no children;
    - isRoot();// if it is the root of the Conditional Tree
    - evaluate();// evaluate the whole condition.

The structure above had been implemented and the javadoc is done:
https://github.com/Vankar/zest
Now test.
Assumption: 2 Expression of the same class can have the same name!

July 7th

Modifying ZestAssertion
some refractoring
some changes in ZestConditional (implements now ZestExpressionElement)
Some compilation errors using the current ZestPrinter.

July 9th-12th

Changed Repo location:
https://github.com/seccoale/zest
making some changes suggested by the mentor to the code;

these the comments and the Todos:

1. [DONE] ZestExpression.deepCopy is still 'TODO' - you'll need to implement that before you check things in as ZAP makes heavy use of this method
2. The ZestEpression* methods that have been copied include implementations of deepCopy that will need to change - eg the code for copying if and elseStatements will be implemented in ZestExpression
3. [DONE] ZestAssert* (apart from ZestAssertion of course) can all be deleted as they are effectively replaced by ZestAssertion + the relevant ZestExpression
4. [DONE] You can improve the efficiency of both ZestExpressionAnd and ZestExpressionOr, for example if just one of the child expressions of an Or is true then the whole expression will be true (although you'll need to take into account isInverse;)
5. [DONE] Does ZestExpression really need to keep track of its parent? I dont think any of the other

ZestElements do and it would make things easier if it doesnt
6. [DONE] I'm not keen on the DEFAULT_NAMEs - right now none of the other elements have these. We may change that, but not using these names, so best to remove them ;)
7. [DONE/TO CHECK] ZestPrinter doesnt seem to handle complex expressions, although I might have missed that
8. [DONE]ZestConditional.deepCopy will need to make deep copies of the rootExpression, the ifStatements and the elseStatements, eg similar to the way ZestExpressionURL does now. Basically in a deepCopy only _non_ ZestElements can be directly copies. For all ZestElements you must use deepCopy.
9. [DONE]ZestAssertion will need a deepCopy
10. All new classes should include the 'standard' Mozilla Public License header (eg ZestExpression etc).

thinking about the creation of a new Class ZestStructuredExpression which extends ZestExpression and extended by ZestExpression[And,Or]. [implemented]

July 13th

Reviewing the code and test.
Maybe the methods isRoot & isLeaf can be deleted.
Removed isRoot & setRoot methods.
Created methods for removing children from a StructuredExpression
Changed inheritance of some classes, and modified some deepCopy() methods.
Editing current tests.
All current tests passed!
Creating new tests for ZestStructuredExpression.----> [DONE]
Fixed some bugs raised by the tests.

July 14th

Continuing with other tests: One unit test each type of Expression.

- ExpressionLength: Coverage 100%

Edit ZestExpressionURL.deepCopy method.

- StructuredExpression: Coverage 100%

Merge with current mozilla/zest repo

July 15th

Tested Lazy evaluation for both ZestExpressionAnd & ZestExpressionOr
Class ZestExpressionResponseTime: the flag “greaterThan” could be completely replaced by “isInverse”. Keeping the default:

- Response.getResponseTime() <= Expression.getResponseTime();

Testing class ZestExpressionURL.

pull request

Start thinking of ZestLoop
Design phase avaible at this link

July 16th

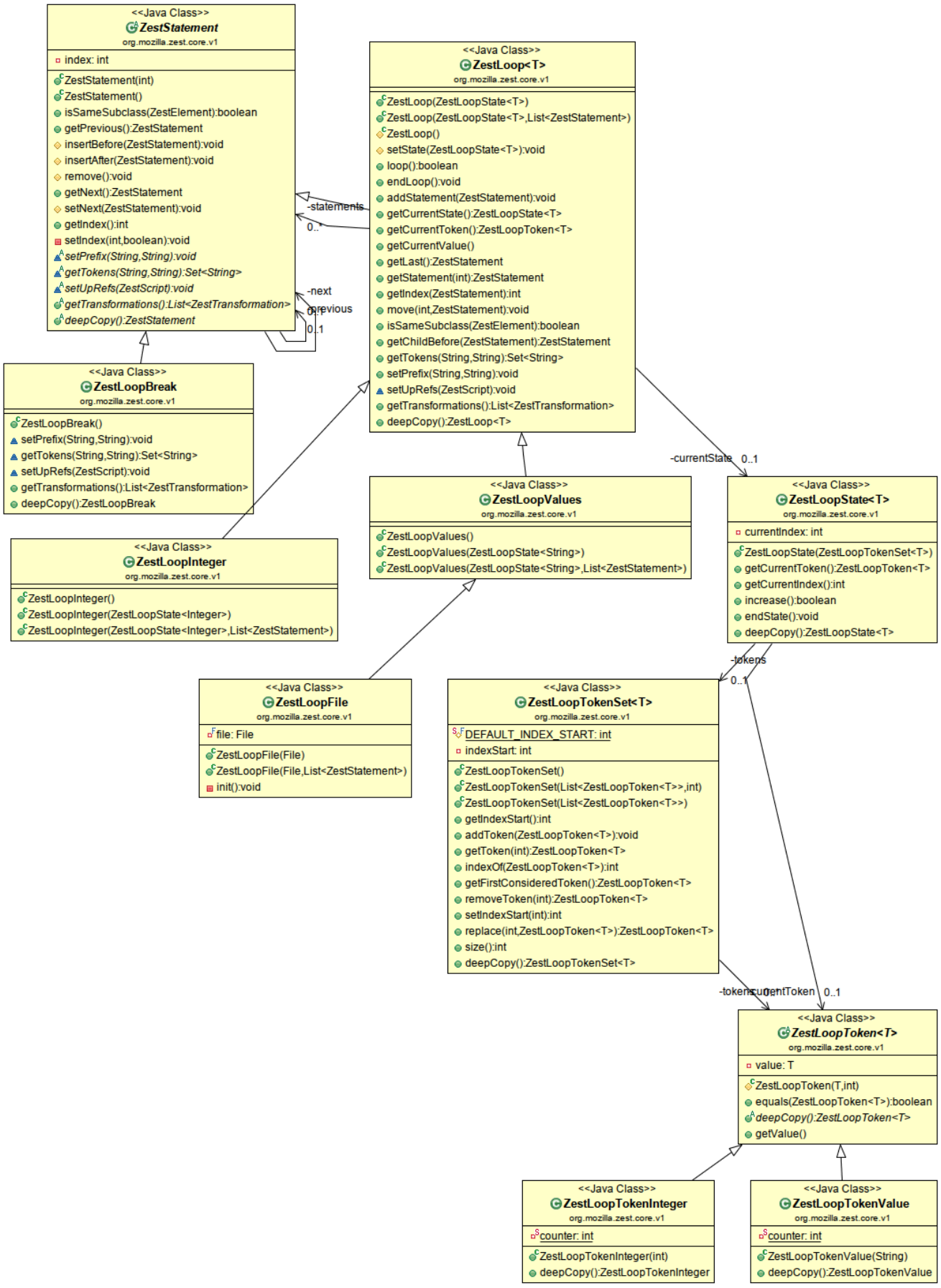
Received comments from the pull request, changed and committed!
Code had been succesfully merged!

July 17th-18th

Started working on ZestLoop. First code and first UML

July 21st

Continuing on ZestLoop after feedback from the mentor.
current UML for Loops: next page.


<<Java Class>>
ZestStatement
org.mozilla.zest.core.v1
index: int
ZestStatement(int)
ZestStatement()
isSameSubclass(ZestElement):boolean
getPrevious():ZestStatement
insertBefore(ZestStatement):void
insertAfter(ZestStatement):void
remove():void
getNext():ZestStatement
setNext(ZestStatement):void
getIndex():int
setIndex(int,boolean):void
setPrefix(String,String):void
getTokens(String,String):Set<String>
setUpRefs(ZestScript):void
getTransformations():List<ZestTransformation>
deepCopy():ZestStatement
-statements
0..*
-next
-previous
0..1
<<Java Class>>
ZestLoop<T>
org.mozilla.zest.core.v1
ZestLoop(ZestLoopState<T>)
ZestLoop(ZestLoopState<T>,List<ZestStatement>)
ZestLoop()
setState(ZestLoopState<T>):void
loop():boolean
endLoop():void
addStatement(ZestStatement):void
getCurrentState():ZestLoopState<T>
getCurrentToken():ZestLoopToken<T>
getCurrentValue()
getLast():ZestStatement
getStatement(int):ZestStatement
getIndex(ZestStatement):int
move(int,ZestStatement):void
isSameSubclass(ZestElement):boolean
getChildBefore(ZestStatement):ZestStatement
getTokens(String,String):Set<String>
setPrefix(String,String):void
setUpRefs(ZestScript):void
getTransformations():List<ZestTransformation>
deepCopy():ZestLoop<T>
-currentState
0..1
<<Java Class>>
ZestLoopBreak
org.mozilla.zest.core.v1
ZestLoopBreak()
setPrefix(String,String):void
getTokens(String,String):Set<String>
setUpRefs(ZestScript):void
getTransformations():List<ZestTransformation>
deepCopy():ZestLoopBreak
<<Java Class>>
ZestLoopValues
org.mozilla.zest.core.v1
ZestLoopValues()
ZestLoopValues(ZestLoopState<String>)
ZestLoopValues(ZestLoopState<String>,List<ZestStatement>)
<<Java Class>>
ZestLoopState<T>
org.mozilla.zest.core.v1
currentIndex: int
ZestLoopState(ZestLoopTokenSet<T>)
getCurrentToken():ZestLoopToken<T>
getCurrentIndex():int
increase():boolean
endState():void
deepCopy():ZestLoopState<T>
<<Java Class>>
ZestLoopInteger
org.mozilla.zest.core.v1
ZestLoopInteger()
ZestLoopInteger(ZestLoopState<Integer>)
ZestLoopInteger(ZestLoopState<Integer>,List<ZestStatement>)
-tokens
0..1
<<Java Class>>
ZestLoopFile
org.mozilla.zest.core.v1
file: File
ZestLoopFile(File)
ZestLoopFile(File,List<ZestStatement>)
init():void
<<Java Class>>
ZestLoopTokenSet<T>
org.mozilla.zest.core.v1
DEFAULT_INDEX_START: int
indexStart: int
ZestLoopTokenSet()
ZestLoopTokenSet(List<ZestLoopToken<T>>,int)
ZestLoopTokenSet(List<ZestLoopToken<T>>)
getIndexStart():int
addToken(ZestLoopToken<T>):void
getToken(int):ZestLoopToken<T>
indexOf(ZestLoopToken<T>):int
getFirstConsideredToken():ZestLoopToken<T>
removeToken(int):ZestLoopToken<T>
setIndexStart(int):int
replace(int,ZestLoopToken<T>):ZestLoopToken<T>
size():int
deepCopy():ZestLoopTokenSet<T>
-tokens
-currentToken
0..1
<<Java Class>>
ZestLoopToken<T>
org.mozilla.zest.core.v1
value: T
ZestLoopToken(T,int)
equals(ZestLoopToken<T>):boolean
deepCopy():ZestLoopToken<T>
getValue()
<<Java Class>>
ZestLoopTokenInteger
org.mozilla.zest.core.v1
counter: int
ZestLoopTokenInteger(int)
deepCopy():ZestLoopTokenInteger
<<Java Class>>
ZestLoopTokenValue
org.mozilla.zest.core.v1
counter: int
ZestLoopTokenValue(String)
deepCopy():ZestLoopTokenValue

July 22nd

After a tip from the mentor I tested the current code serializing and deserializing it with GSON. I encountered some problems using generics. I fixed creating some other classes which extended ZestLoop<T>. In this way there is no code duplication and everything's working fine!
Next step: test and improvements

July 23rd

Testing ZestLoopToken*, ZestLoopState and all subclasses.
Tested all subclasses. Missing ZestLoop[Integer,String,File].

July 24th

The usage of generics can create some problems using gson. It will not if the whole Loop is considered. I think that this situation should not create problems because ZestLoopToken, ZestLoopTokenSet & ZestLoopTokenState has no sense disconnected from their associated loop.

July 25th-28th

Fixing problems related to the usage of generics.
Removing some useless features.
Changing design & development.
Testing.

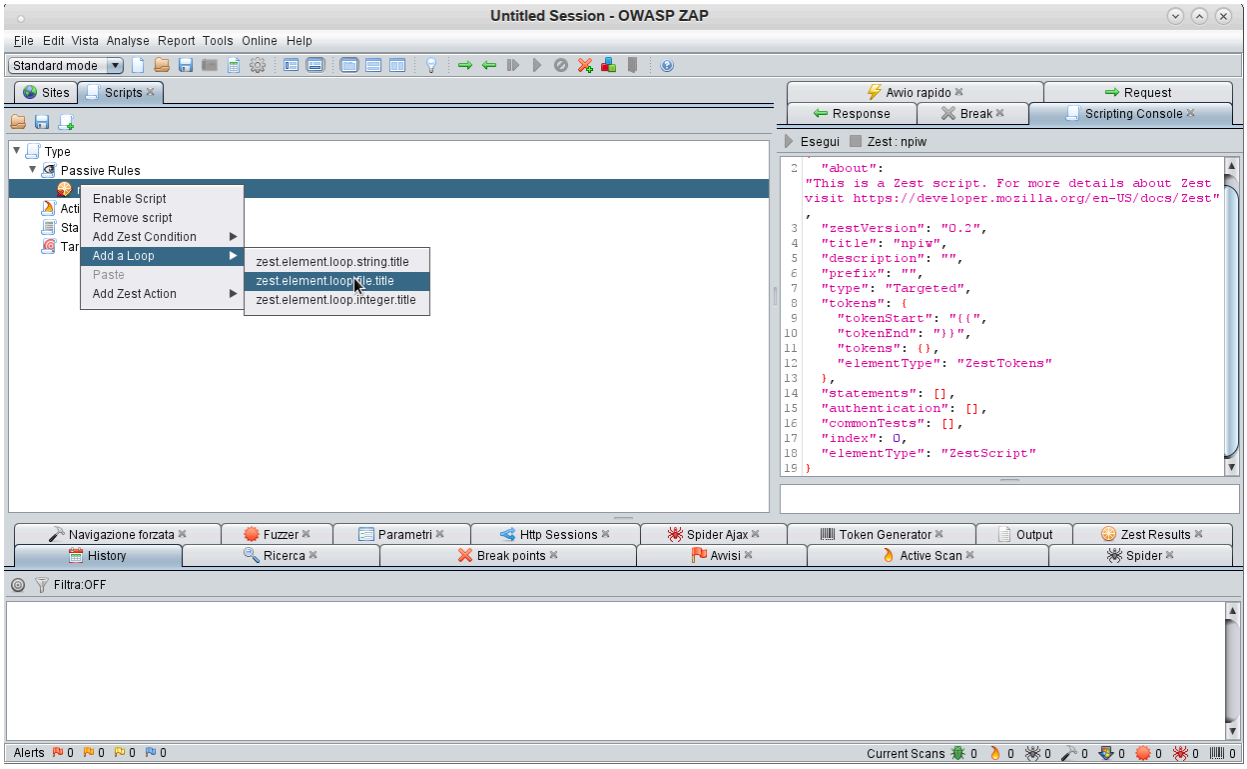
Current UML

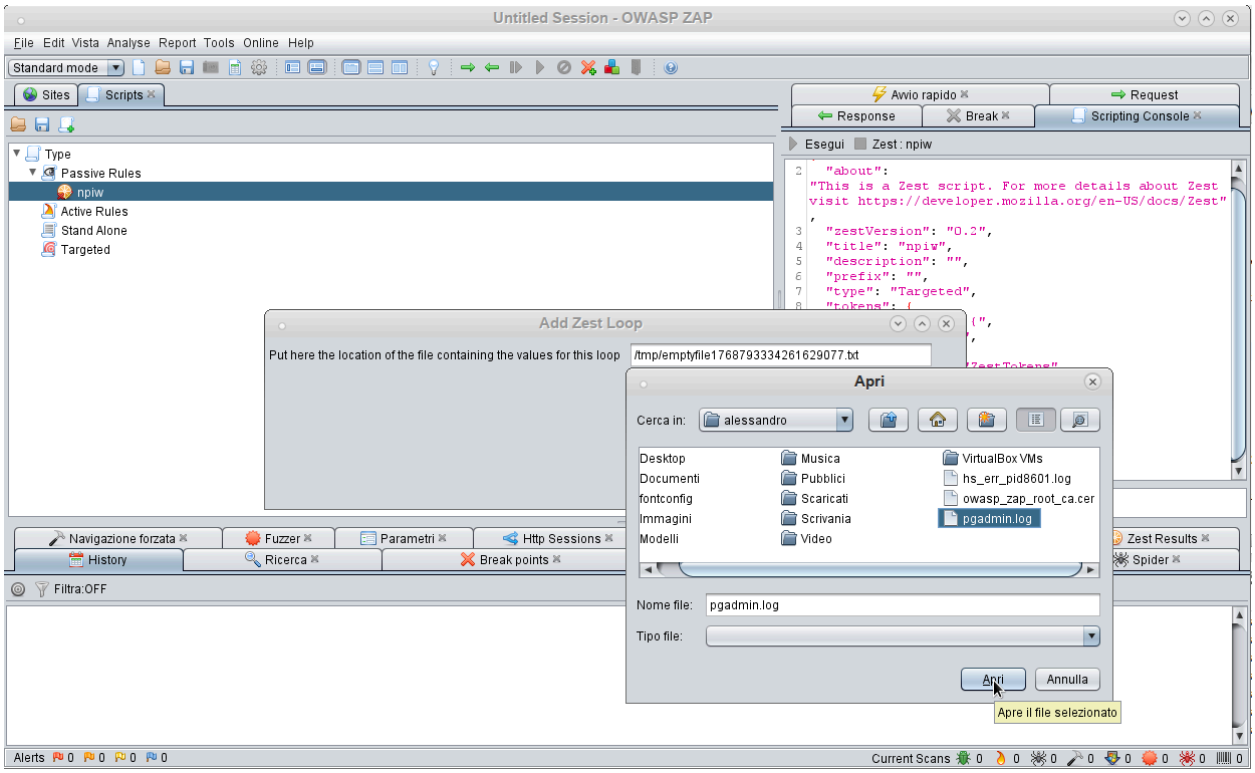
July 29th- August 5th

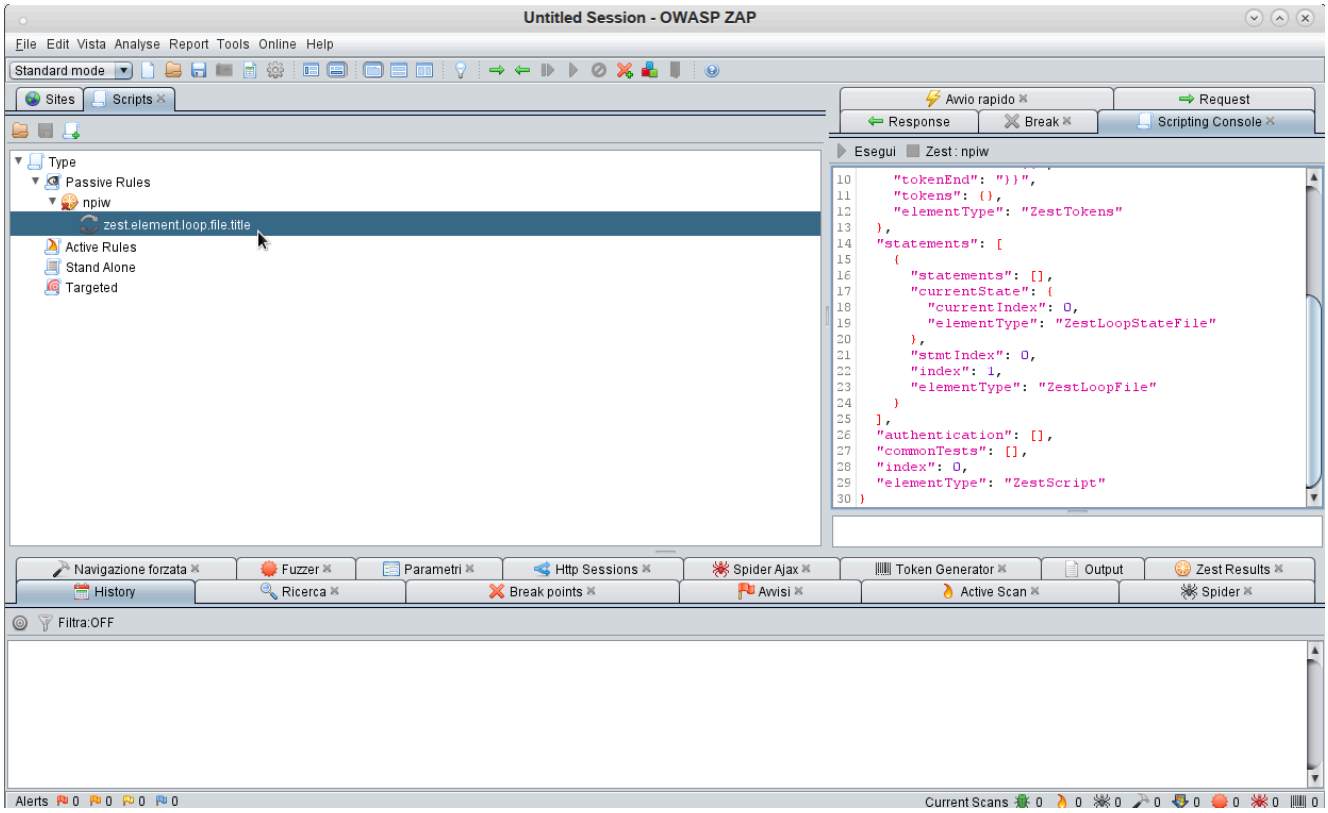
Many things:

- Completed ZestLoops, tested and merged;
- Working on OWASP ZAP UI for loops first and for Complex Conditionals later.

Here some screenshots of Loops UI:


Untitled Session - OWASP ZAP
File Edit Vista Analyse Report Tools Online Help
Standard mode
Sites
Scripts
Type
Passive Rules
Enable Script
Remove script
Add Zest Condition
Add a Loop
Paste
Add Zest Action
zest.element.loop.string.title
zest.element.loop.file.title
zest.element.loop.integer.title
Avvio rapido
Request
Response
Break
Scripting Console
Esegui
Zest : npiw
"about":
"This is a Zest script. For more details about Zest visit https://developer.mozilla.org/en-US/docs/Zest"
"zestVersion": "0.2",
"title": "npiw",
"description": "",
"prefix": "",
"type": "Targeted",
"tokens": {
"tokenStart": "{{",
"tokenEnd": "}}",
"tokens": {},
"elementType": "ZestTokens"
},
"statements": [],
"authentication": [],
"commonTests": [],
"index": 0,
"elementType": "ZestScript"
Navigazione forzata
Fuzzer
Parametri
Http Sessions
Spider Ajax
Token Generator
Output
Zest Results
History
Ricerca
Break points
Avvisi
Active Scan
Spider
Filtra:OFF
Alerts
Current Scans


Untitled Session - OWASP ZAP
File Edit Vista Analyse Report Tools Online Help
Standard mode
Sites
Scripts
Type
Passive Rules
npiw
Active Rules
Stand Alone
Targeted
Add Zest Loop
Put here the location of the file containing the values for this loop
/tmp/emptyfile1768793334261629077.txt
Apri
Cerca in:
alessandro
Desktop
Documenti
fontconfig
Immagini
Modelli
Musica
Pubblici
Scaricati
Scrivania
Video
VirtualBox VMs
hs_err_pid8601.log
owasp_zap_root_ca.cer
pgadmin.log
Nome file:
pgadmin.log
Tipo file:
Apri
Annulla
Apre il file selezionato
Avvio rapido
Request
Response
Break
Scripting Console
Esegui
Zest : npiw
"about":
"This is a Zest script. For more details about Zest visit https://developer.mozilla.org/en-US/docs/Zest"
"zestVersion": "0.2",
"title": "npiw",
"description": "",
"prefix": "",
"type": "Targeted",
Navigazione forzata
Fuzzer
Parametri
Http Sessions
Zest Results
History
Ricerca
Break points
Spider
Filtra:OFF
Alerts
Current Scans



Same things works also for LoopInteger and LoopString.