CODE PULSE

Technical Task Report

Author
AmirMahdi Kousheshi

Contents

1 Open source projects

For this task, I used some open source java projects like Mars, WebGoat, Jenkins, openmrs, Qcadoo, PetClinic and etc. I also faced several challenges that I explain below.

In summary, I read CodePulse's paper to get to know about this tool and why this tool matters for penetration tests and real-time code coverage. Then I cloned some open source projects and import them on CodePulse and work with them to get some results. I also thought about the features that would improve this tool and wrote a simple program based on them.

1.1 Mars

Mars was the first tool that I worked with. This tool is an emulator for MIPS assembly that allows us to write a program with MIPS and run it. It also has many features for debugging and monitoring the registers. MIPS is published by Missouri state university.

1.2 WebGoat

WebGoat is a deliberately insecure web application maintained by OWASP designed to teach web application security lessons.

1.3 Jenkins

Jenkins is an open source automation server. It helps with automating the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

1.4 OpenMRS

OpenMRS is a collaborative open-source project to develop software to support the delivery of health care in developing countries.

1.5 Qcadoo

Qcadoo MES is an Internet application for production management targeted at small and medium companies.

1.6 PetClinic

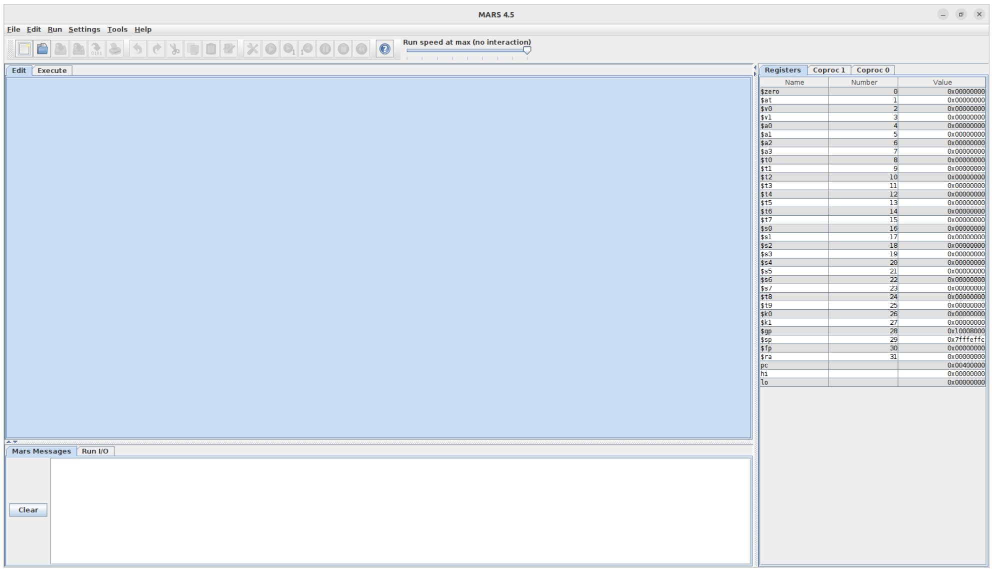
The Spring PetClinic is a simple application designed to show how the Spring stack can be used to build simple, but powerful database-oriented applications.

2 Using CodePulse

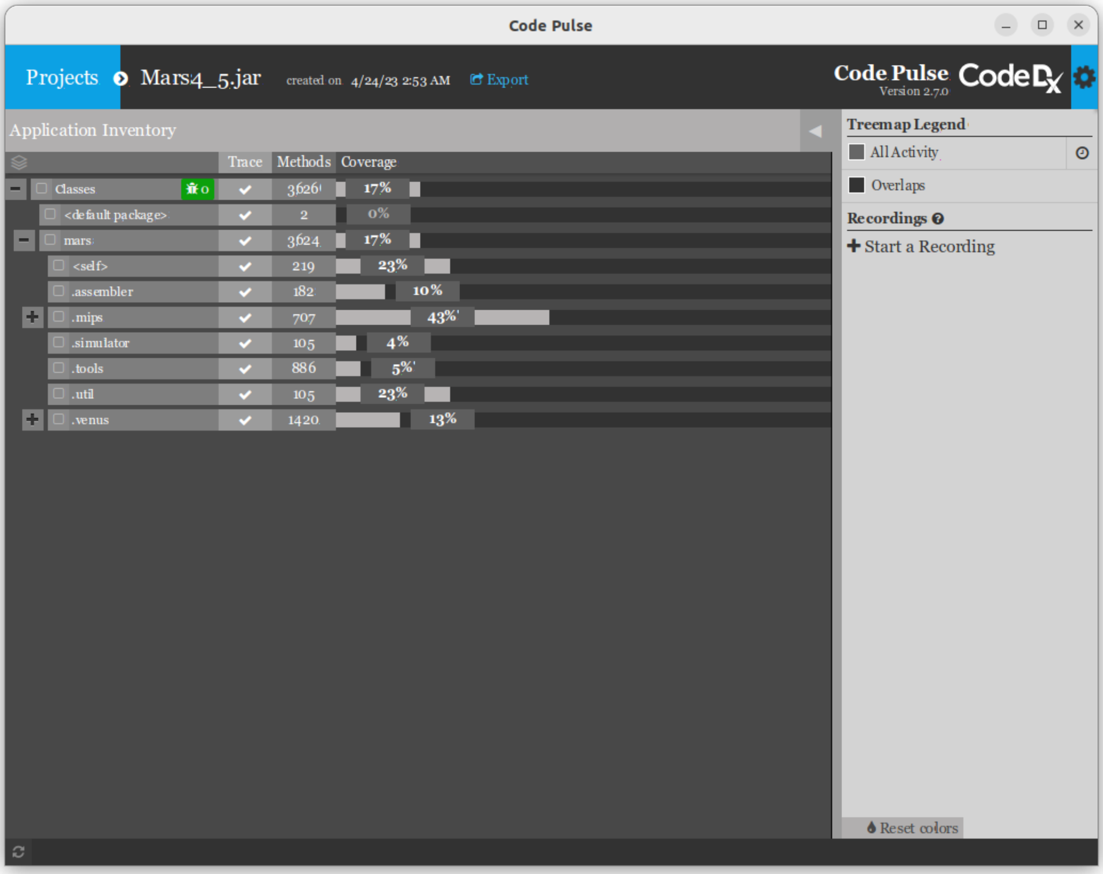
After using CodePulse on these projects, I learned and also found many things. I learned many useful things about penetration test and real-time code coverage. First, I used Mars and import it on CodePulse.

2.1 Mars

Honestly, Mars was the simplest tool that I tried to use, and was so easy and smooth to import on CodePulse and to get results. It worked with Java 11 which was necessary for CodePulse. Here are some images of Mars and the results of using it on CodePulse.

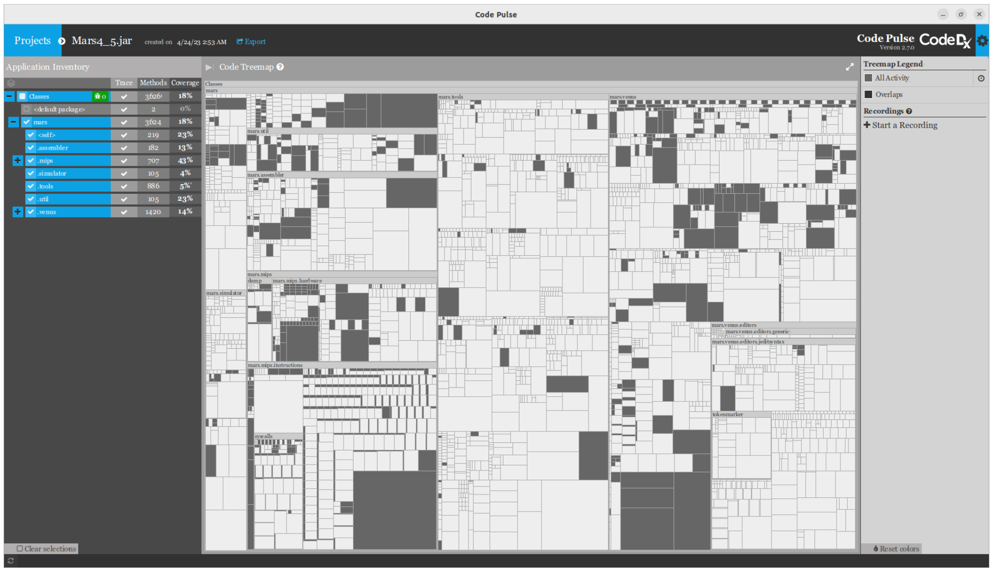

MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

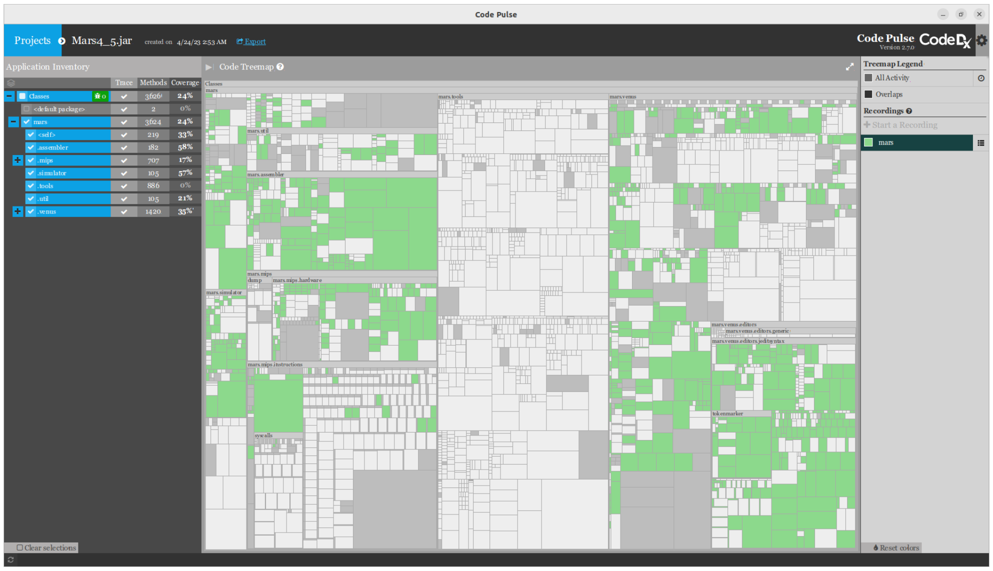


Mars4_5.jar created on 4/24/23 2:53 AM

	Trace	Methods	Coverage
Classes 0	✓	3626	17%
<default package>	✓	2	0%
mars	✓	3624	17%
<self>	✓	219	23%
.assembler	✓	182	10%
.mips	✓	707	43%
.simulator	✓	105	4%
.tools	✓	886	5%
.util	✓	105	23%
.venus	✓	1420	13%



	Trace	Methods	Coverage
Classes	✓	3626	18%
<default package>	✓	2	0%
mars	✓	3624	18%
<self>	✓	219	23%
.assembler	✓	182	13%
.mips	✓	707	43%
.simulator	✓	105	4%
.tools	✓	886	5%
.util	✓	105	23%
.venus	✓	1420	14%

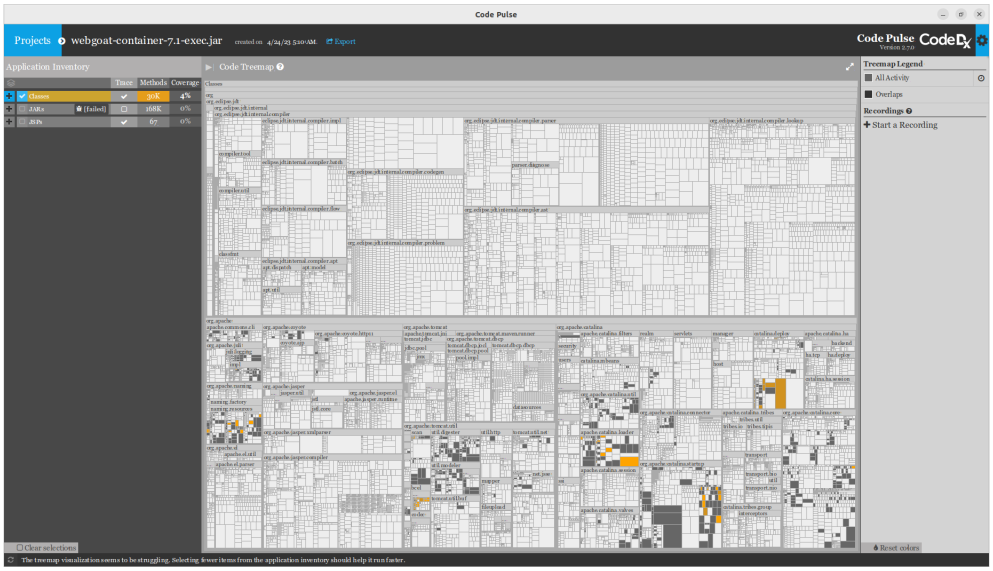


	Trace	Methods	Coverage
Classes	✓	3626	24%
<default package>	✓	2	0%
mars	✓	3624	24%
<self>	✓	219	33%
.assembler	✓	182	58%
.mips	✓	707	17%
.simulator	✓	105	57%
.tools	✓	886	0%
.util	✓	105	21%
.venus	✓	1420	35%

For Mars, I opened a file and wrote a MIPS program and compiled and ran it. The last picture is the result of this workflow.

2.2 WebGoat

For WebGoat, I downloaded the WebGoat container jar file version 7.1 and import on CodePulse. This result is produced by me manually clicking on some buttons and reading a lesson.



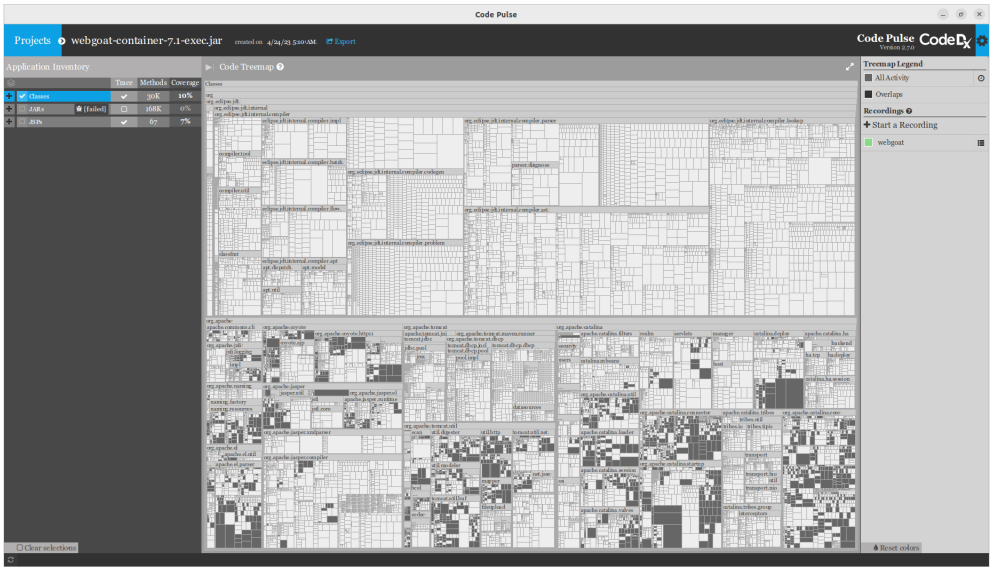
	Trace	Methods	Coverage
☑ Classes	✓	30K	4%
□ JARs [failed]	□	168K	0%
□ JSPs	✓	67	0%

- ▨ All Activity
- ■ Overlaps

Recordings

+ Start a Recording

The treemap visualization seems to be struggling. Selecting fewer items from the application inventory should help it run faster.



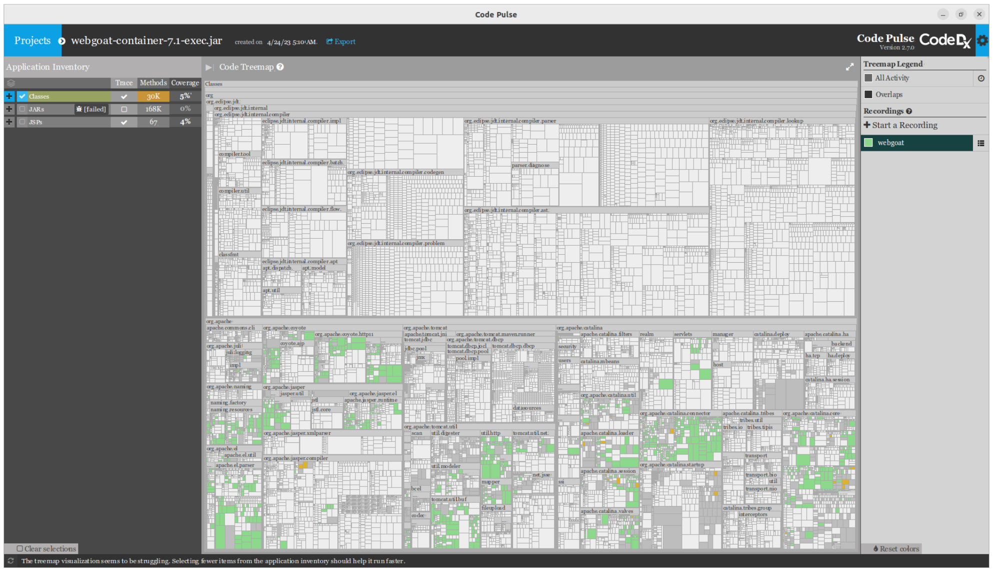
	Trace	Methods	Coverage
☑ Classes	✓	30K	10%
□ JARs [failed]	□	168K	0%
□ JSPs	✓	67	7%

- ▨ All Activity
- ■ Overlaps

Recordings

+ Start a Recording

- 🟩 webgoat



webgoat-container-7.1-exec.jar created on 4/24/23 5:10 AM. Export

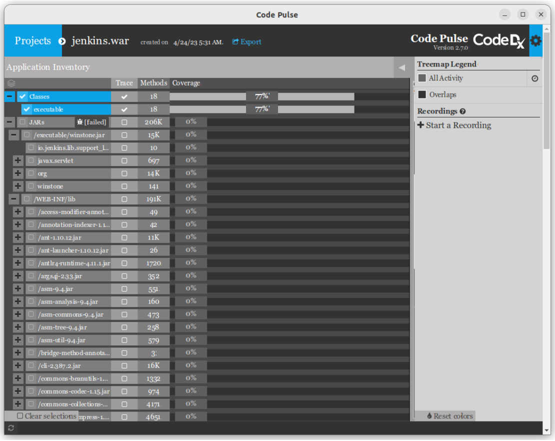
Application Inventory

	Trace	Methods	Coverage
☑ Classes	✓	30K	5%
□ JARs [failed]	□	168K	0%
□ JSPs	✓	67	4%

The treemap visualization seems to be struggling. Selecting fewer items from the application inventory should help it run faster.

2.3 JenKins

For jenkins, I downloaded the web application resource file (war) and import it on codepulse. There was a same issue that I faced for this projects (jenkins, qcadoo, openmrs), and it was the dispatcher for this projects. This projects use a dispatcher and they can do their work with tomcat but when I ran these projects based on codepulse java agent, they could not use tomcat to use dispatchers.



jenkins.war created on 4/24/23 5:31 AM. Export

Application Inventory

	Trace	Methods	Coverage
☑ Classes	✓	18	77%
☑ executable	✓	18	77%
□ JARs [failed]	□	206K	0%
□ /executable/winstone.jar	□	15K	0%
□ io.jenkins.lib.support_l...	□	10	0%
□ javax.servlet	□	697	0%
□ org	□	14K	0%
□ winstone	□	141	0%
□ /WEB-INF/lib	□	191K	0%
□ /access-modifier-annot...	□	49	0%
□ /annotation-indexer-1.1...	□	42	0%
□ /ant-1.10.12.jar	□	11K	0%
□ /ant-launcher-1.10.12.jar	□	26	0%
□ /antlr4-runtime-4.11.1.jar	□	1720	0%
□ /args4j-2.33.jar	□	352	0%
□ /asm-9.4.jar	□	551	0%
□ /asm-analysis-9.4.jar	□	160	0%
□ /asm-commons-9.4.jar	□	473	0%
□ /asm-tree-9.4.jar	□	258	0%
□ /asm-util-9.4.jar	□	579	0%
□ /bridge-method-annota...	□	3	0%
□ /cli-2.387.2.jar	□	16K	0%
□ /commons-beanutils-1...	□	1332	0%
□ /commons-codec-1.15.jar	□	974	0%
□ /commons-collections-...	□	4171	0%
mpress-1....	□	4651	0%

3 Strengths and the opportunities to Improve

After working with CodePulse, I discovered some opportunities that would be worth exploring. I bring them here:

3.1 Speed

The first thing we deal with is the speed of CodePulse. In some projects that are heavy (like Jenkins), the speed of CodePulse is lower than lighter projects (like Mars). When I was reading CodePulse code on its GitHub, I found a way that could improve its speed. When a project is run based on CodePulse server (here is a java agent), after clicking on each item, a data will be send to CodePulse and it shows the results for us. If CodePulse batches the data in some requests and then visualizes them to show the results, it will make the target's execution faster and will not affect CodePulse real-time promises as well. For example when we use a web application, it can slice the data that it receives from client and then batches them and visual them.

3.2 Java 11

The important thing that we deal with, is Java 11. As CodePulse mentions on its GitHub document, it can only work with projects that use java version 11, while many recent projects use a newer version of java like 17 or 19. If the CodePulse could import them it would have a broader use. There were many projects that I could not load on CodePulse because they used java 17 or 19. I believe CodePulse uses some features that are deprecated in newer versions of java. Also the specific tools that added in newer versions of java could be added CodePulse and make this tool more powerful.

3.3 Visualize Data

As the paper stated, visualizing the data is so important. The data (almost functions) have green and grey and yellow colors. In CodePulse, after recording a workflow, we can see which functions are called. It is important for developers after a penetration tests to have more details about functions. We can add a feature to this tool so that the functions that are called more times have a darker color than the functions that are called less often. It can more or less show that these function are more tested than the second group of functions. It can also show that the first group of functions have probably more line coverage testing than the other group and it can be a good data for developers who want to find the bugs.

3.4 DAST Data

The important thing as we saw on the paper was the DAST tools. These security testing tools will show us the data that they use to test our application. But sometimes we do not have the data. So if a feature was added to CodePulse to show after an attack, which data causes the bug in which functions, the developers could easier debug the code and find out why the attack happened. For example if a database was added to CodePulse so that it'd log which functions called with which arguments, it would be so good and give more data about bugs.

3.5 Code Coverage

After recording a project, CodePulse can also analyze the coverage data and bring the developers more useful data. One example of such data could be code coverage. For java there are many things that can run tests (unit tests, integration test and etc.) and show developers which functions had been called more in a recording of CodePulse that have few unit tests. For example after a recording of Mars, CodePulse can run the Mars unit tests with maven or something else and provide some data like "the function X has several calls in this recording but its code coverage or the number of unit tests written for this functions are not enough". In this case developers would know which essential functions are in use that are not tested, because this functions could be the reason of an attack.

3.6 Parallel codepulse

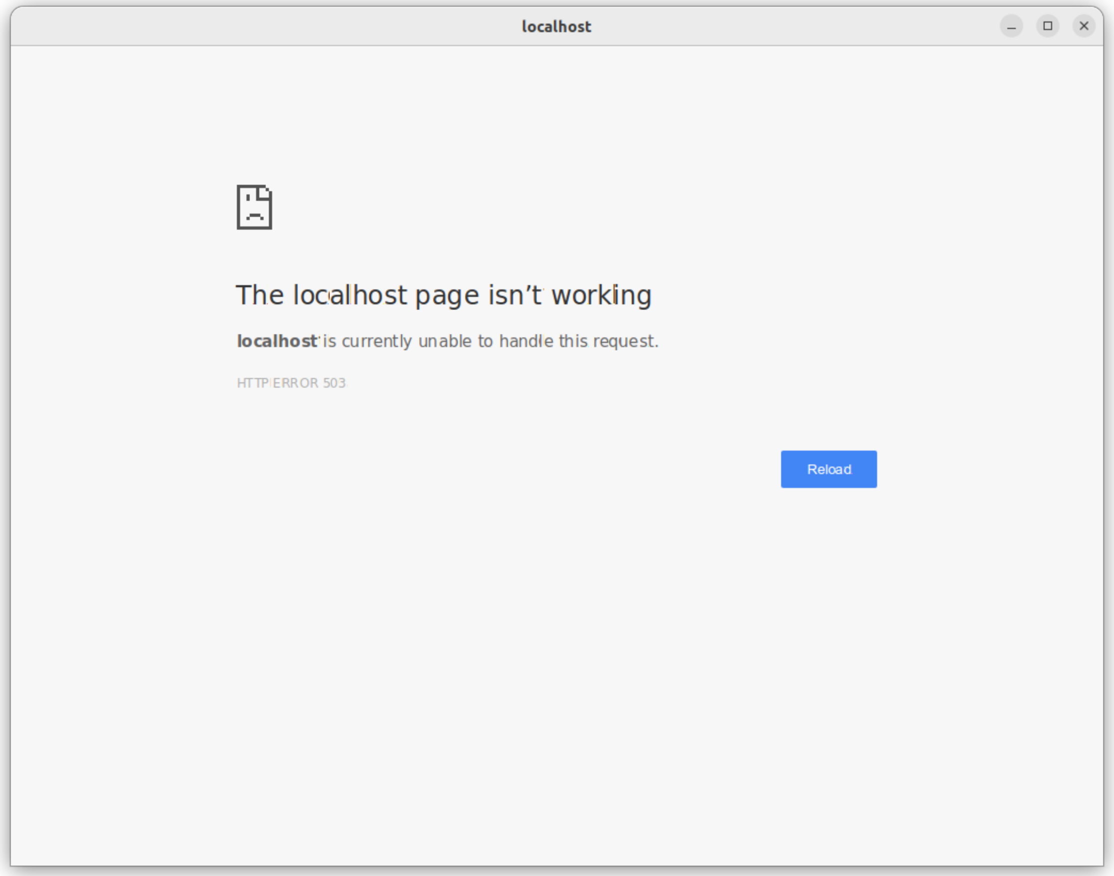
As an feature, we can do something that codepulse run the projects parallel and collect data from them and send this data to the main client that visualize the data. For example when we ran three projects parallel, (in three different docker containers)it could receive data from this three projects and visual them on a main codepulse client and then it could show the total coverage of this three projects and also show the coverage of each project separately.

4 The Challenges I Faced

After installing CodePulse, I faced many challenges. At first, I ran CodePulse but it didn't lunch. Then I ran the "./CodePulse –log" to check what is the problem. Then I found out the directory that I cloned CodePulse on, is "code pulse", and in CodePulse log it logged the java error I.O exception "file not found". After I removed the space and rename the directory to "CodePulse" it worked well. So I think this is a bug in its code. I will report it on GitHub.

The second challenge was with the applications that were using a version of Java that was newer than version 11. That was so bad because most applications use a newer version than 11 and I could not import them on CodePulse.

At first when I installed CodePulse, I was using a proxy (because of the filtering of Internet in my country) and CodePulse wasn't loading, it was like the picture below.



localhost

The localhost page isn’t working

localhost is currently unable to handle this request.

HTTP ERROR 503

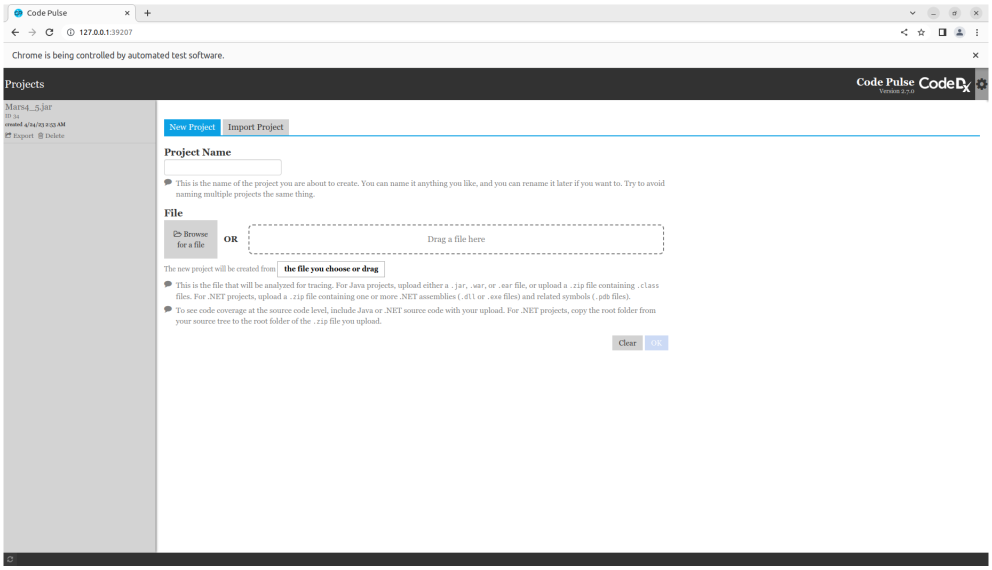
At the beginning I did not know what is happening and I did not aware this is because of using proxy. I did some search and suddenly I disconnected with my proxy and then CodePulse lunched well. So I aware that my proxy did not exclude the local host and because of this CodePulse can not run as well. So it brings me an idea that I can run CodePulse on web and somehow being portable CodePulse.

It was also same problems with some application like jenkins, qcadoo and openmrs. This projects use a disptcher for their URLs. In a normal way, they use tomcat for their dispatcher, but here when I ran these projects based on codepulse java agent, they could not use dispatcher. Also some projects like petclinic, use some applications like jenkins, but when the ran based on codepulse java agent, they could not use it and throw exceptions. It was also a big problem that i faced and could not solve it.

Another problem I faced was that codepulse will stop suddenly and won’t work. I found this situations when I import some projects and remove them and import the other projects. Also this bug occurred when we start a recording and stop it and export its data. In this situation it will crash and won’t work and I had to kill its process from monitoring system.

5 CodePulse on web

After I the challenges with proxy and CodePulse, I found out I can run CodePulse on web. When CodePulse wasn't running, I found its process id and its port and then on chrome I searched "localhost:port" and CodePulse opened on chrome. So I wrote a program that can run CodePulse and after that use netstat to find its port and then open CodePulse on web. I used selenium for this program and they are some features like running a project automatically and recording a usage of that project. In other words somehow this program is an automated CodePulse. It can run CodePulse, import a project, run the project based on CodePulse java agent and start recording and there is no need for a person to do anything. There are also some features that can be added to this program like automatically running a DAST tool which can attack the project and also export and analyzing the result of CodePulse and beings them for developers that I want to add to this program. The program code is in repository. Some images of the program's final result are included here.



Chrome is being controlled by automated test software.

Mars4_5.jar
ID 34
created 4/24/23 2:53 AM

Project Name

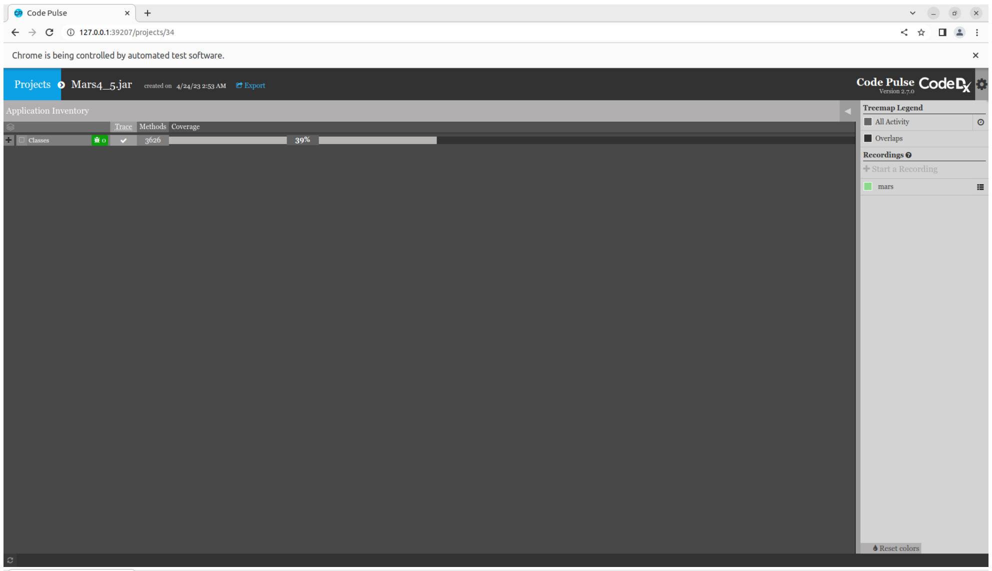
This is the name of the project you are about to create. You can name it anything you like, and you can rename it later if you want to. Try to avoid naming multiple projects the same thing.

File

The new project will be created from **the file you choose or drag**

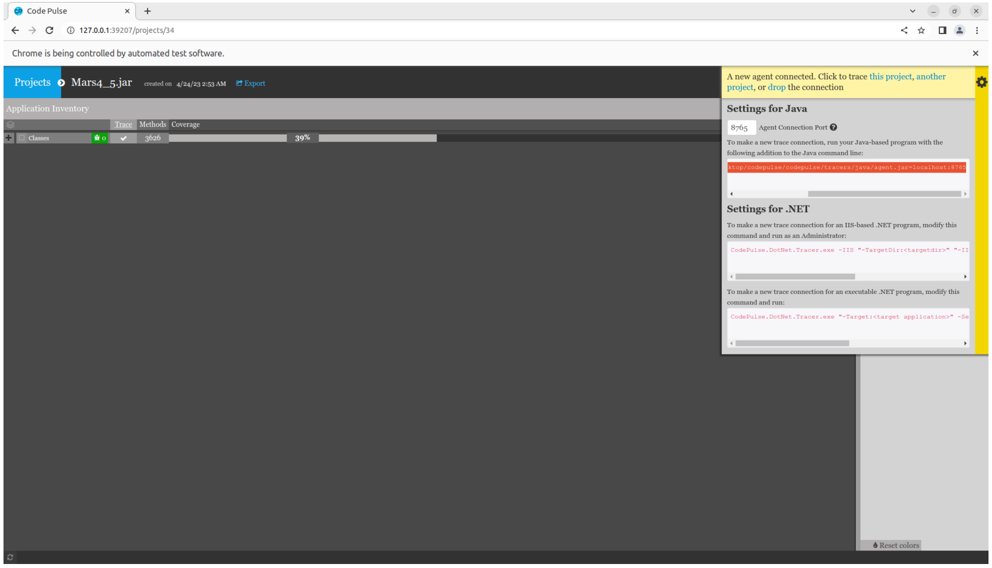
This is the file that will be analyzed for tracing. For Java projects, upload either a `.jar`, `.war`, or `.ear` file, or upload a `.zip` file containing `.class` files. For .NET projects, upload a `.zip` file containing one or more .NET assemblies (`.dll` or `.exe` files) and related symbols (`.pdb` files).

To see code coverage at the source code level, include Java or .NET source code with your upload. For .NET projects, copy the root folder from your source tree to the root folder of the `.zip` file you upload.



Chrome is being controlled by automated test software.

	Trace	Methods	Coverage
Classes 0	✓	3626	39%



Chrome is being controlled by automated test software.

	Trace	Methods	Coverage
Classes 0	✓	3626	39%

A new agent connected. Click to trace this project, another project, or drop the connection

Settings for Java

8765 Agent Connection Port

To make a new trace connection, run your Java-based program with the following addition to the Java command line:

```
ktop/codepulse/codepulse/tracers/java/agent.jar=localhost:8765
```

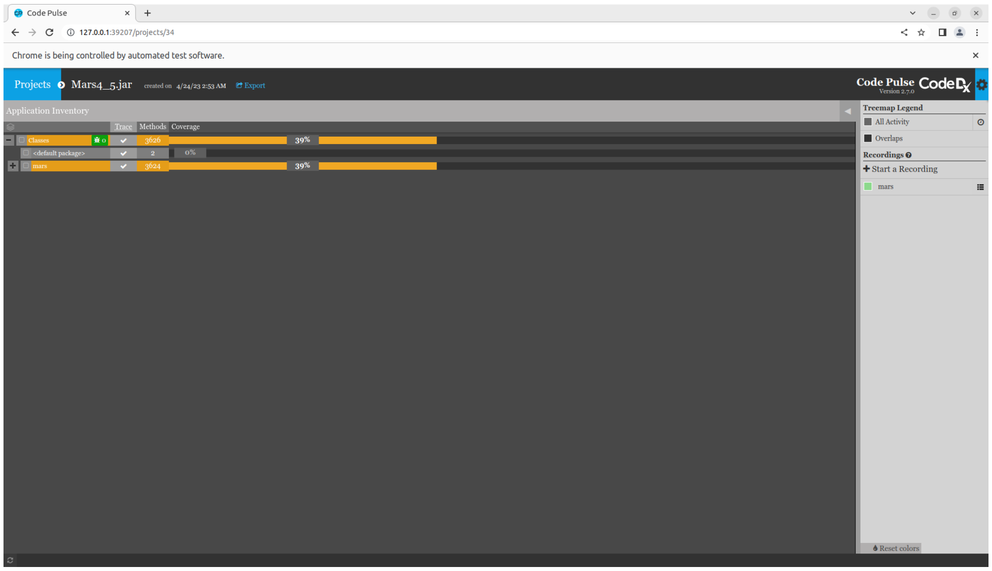
Settings for .NET

To make a new trace connection for an IIS-based .NET program, modify this command and run as an Administrator:

```
CodePulse.DotNet.Tracer.exe -IIS "-TargetDir:<targetdir>" "-II
```

To make a new trace connection for an executable .NET program, modify this command and run:

```
CodePulse.DotNet.Tracer.exe "-Target:<target application>" -Se
```



Chrome is being controlled by automated test software.

	Trace	Methods	Coverage
Classes 0	✓	3626	39%
<default package>	✓	2	0%
mars	✓	3624	39%

As you see, CodePulse ran on web (chrome) and selenium did its all works (import, ran and etc.).

6 strengths of codepulse

Codepulse is a strong tool in penetration test and real-time code coverage. They also many good and useful features in this tool. Some of this features that I liked and interested on them are visualizing the data and running projects separately from each other and collect data of them. This tool is so good in visualizing the data that it collected from the application. We can find the functions and their usages so easily.

Also wen run codepulse on different ports and import different projects on them and collect data of each one of them.

The recording feature is also so useful that we can start recording coverage of a testing and save it and also open it and analyze the data that it records.