

Humble Avalanche Replay of HTTP Archives

### (Harhar, for short)

“Failing to plan is planning to fail.” Alan Lakein (presumably)

# Introduction

Running performance tests is important and if you're reading this document it means you know that. What is also important is to run *realistic* performance tests. [RFC 3511](http://tools.ietf.org/html/rfc3511) is good, and pre-defined protocol mixes are good too. However there are cases where you want to simply, *precisely* reproduce a real website, web service, or anything HTTP based (like Adaptive Bitrate or SOAP services). There are many reasons why you might want to model one or more of those. What is important is having the ability to do it.

This is what Harhar enables you to do. Harhar stands for “Humble Avalanche Replay of HTTP Archives.” A HTTP archive is, simply put, a JSON (JavaScript Object Notation) file formatted in a pre-defined fashion. It's actually a W3C standard. The specification is available here: <https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/HAR/Overview.html>

A pre-requisite to using this tool is to have good knowledge of Spirent Avalanche.

Harhar is a program that will take one or more HTTP Archive (.har) files, read them, rebuild the requests and responses that were capture and convert that to an Avalanche Action List. In short: you record yourself accessing a website, automatically save all the pages, images and other resources you accessed, and convert that in a way that make it easy to replay in Avalanche – at a thousand fold load.

# 2. Installation

Harhar is available in a .zip archive and doesn't require much of an installation. There are a few .dll to keep in the same folder as harhar.exe. You can get the zip file from this website: <http://alarash.net/harhar>

Harhar has been tested on Windows 7 64 bits, Windows 8 64 bits and on Mono 3.2.8 for Linux (any Linux distribution running this version or above of Mono should work).

## Windows

If you are running a recent version of Windows (Windows Vista or above), you will not have much to do as you likely already have the required .NET files on your computer. If you run into any error while using Harhar, make sure you have the Microsoft .NET 4.5 Framework available here: <http://www.microsoft.com/en-us/download/details.aspx?id=30653>

Download the zip file extract it anywhere you need. You can then run harhar.exe either from the command line (recommended) or by double clicking on it.

## Linux

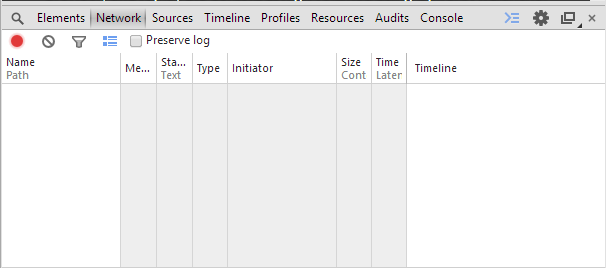
Harhar has been tested on CentOS 6.5 and Ubuntu 12.04. The actual distribution does not really matters. What matters is that you run Mono 3.2+. This is because .NET 4.5 support started in Mono 3.2. The actual version that was tested was Mono 3.2.8. Some distributions will only provide version 2.8 in their official repositories. In such case you’ll need to download and compile Mono 3.2+ from this server: <http://download.mono-project.com/sources/mono/>

# 3. Usage

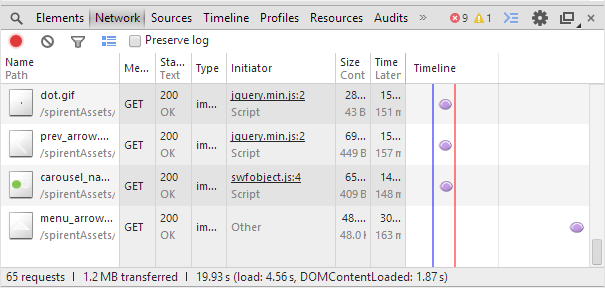
To use Harhar you will need one or more .har files. As of version 1.0.3 only files generated by Google Chrome are fully supported (Firefox versions are in the works).

## Generating .har files

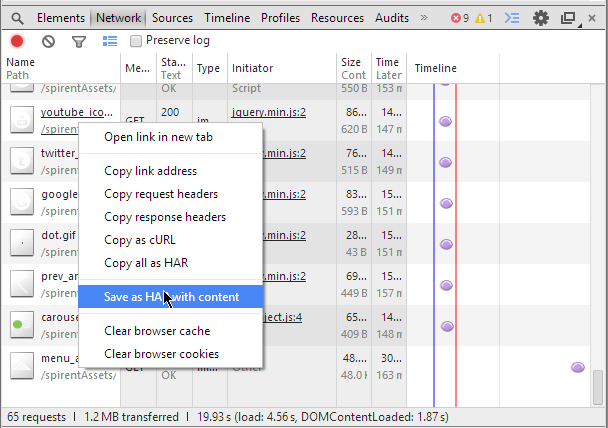
To generate a .har file in Chrome, open the Developer Tools (CTRL + Shift + I) and click on the “Network” tab:



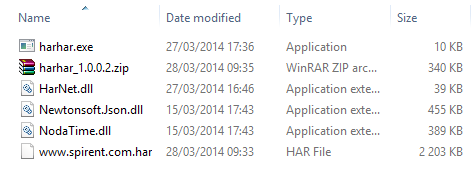
Then simply browse to the web page you want to capture – the resources will show up in the console. For instance, browsing to [http://www.spirent.com](http://www.spirent.com/) :

(Note that there are 9 errors, some of them 404 – Not Found status codes. Harhar will ignore responses that are not in the 200-300 status code range).

Once you have all these resources logged into Chrome, simply right-click and choose “Save as HAR with Content.” The “with Content” part is very important because we use that content to reconstruct the file (so you can put it on the Avalanche web server):

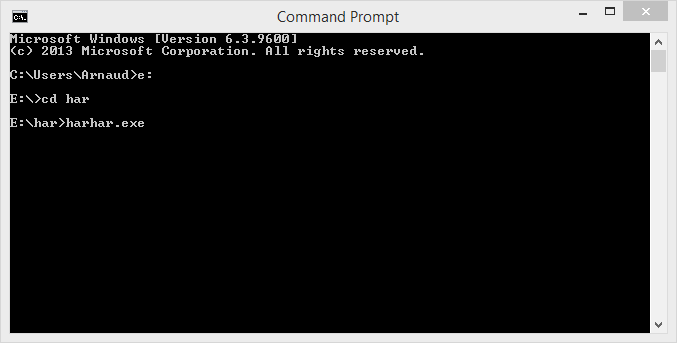


Save that file in some directory of your convenience, for instance E:\har. Then download Harhar and extract it in the same directory, like so:



## Using Harhar

Open a Command Prompt, browse to the directory that holds Harhar and simply execute harhar.exe:



When you invoke Harhar without parameters, it will look for .har files in the current directory (not recursively). If no file is found you will get this message:

No file found in working directory.  
Parsing complete. Press any key to exit.

You can also specify a directory as a parameter of harhar.exe, and Harhar will look into that directory for .har files:

E:\har>harhar.exe e:\har\files  
Total Response size: 1215418 bytes (headers: 17192; bodies: 1198226)  
Total Request size: 28943 bytes (headers: 28943 ; bodies: 0 )  
Found 64 entries in log.  
Creating working directory at E:\har\3-28-2014\_9-43-10\www.spirent.com.har\

If you specify a directory and there is more than one .har file in that directory, Harhar will parse all the files. If you want to force Harhar to parse only one file, specify it explicitly:

E:\har>harhar.exe e:\har\files\www.spirent.com.har  
Total Response size: 1215418 bytes (headers: 17192; bodies: 1198226)  
Total Request size: 28943 bytes (headers: 28943 ; bodies: 0 )  
Found 64 entries in log.

## File hierarchy

When Harhar is executed, a directory is created in the same folder that holds the executable. For instance, if Harhar is in E:\har the folders will be created in that directory. The folder hierarchy is the following:

<root directory holding harhar.exe>  
<execution\_date>  
 <har\_file\_name>  
 <hostnames>  
 <resource\_uri>  
 <resource\_file>

For instance, the file “<http://www.spirent.com/spirentAssets/images/bullets.gif>” that was found in the file [www.spirent.com.har](http://www.spirent.com.har/) and executed on March 28th 2014 at 9h46 min 34s will be stored here:

E:\har\3-28-2014\_9-46-34\www.spirent.com.har\www.spirent.com\spirentAssets\images

We preserve the hierarchy because this way you can re-upload these files as a Transaction Profile in the Avalanche Commander GUI as a “Files from a directory” - and keep the correct hierarchy. This is useful because it keeps the same URLs when replaying the traffic, which might be important for the Devices Under Test.

A few other files are created under the “har\_file\_name” directory: urls.txt and action\_list.txt. The first file is simply a list of all the requests that we parsed (you could conceivably use it with wget or curl for other uses).

The second one is an Avalanche Action List that you can copy/paste into Avalanche Commander. Any header not mandatory in HTTP 1.1 (like Connection: ; Host:, etc..) is included using the “ADDITIONAL\_HEADER” command. The mandatory headers are not added because Avalanche automatically adds them. Be careful with this action list because cookies, being headers, and other headers that might contain authentication.

# 4. Example

For this example we took a HAR file generated from visiting <http://www.spirent.com>.

## Reviewing File Hierarchy

We ran Harhar against the file generated by Chrome, like so:

C:\Users\acastaner\Harhar>harhar.exe [www.spirent.com.har](http://www.spirent.com.har)

This commanded started Harhar, which parsed the specified HAR file and created the following file hierarchy:

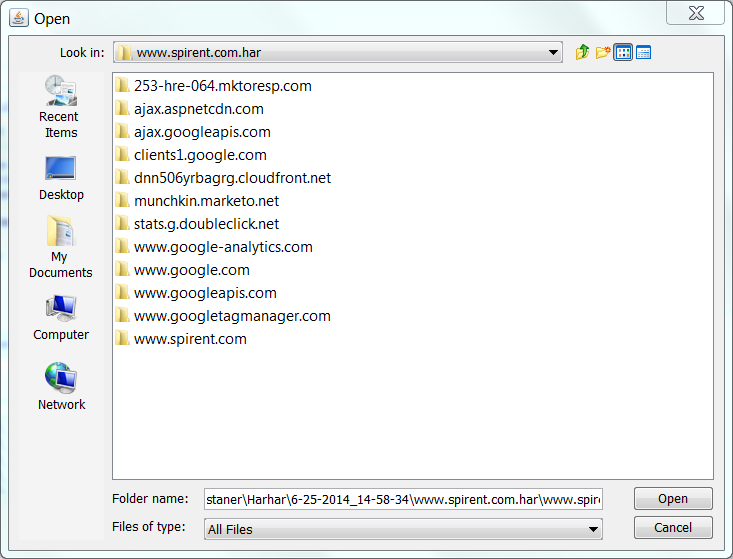
* 6-25-2014\_14-58-34 (because we ran the command on June 25th 2014 at 14h58:34)
  + [www.spirent.com.har](http://www.spirent.com.har) (because that’s the name of the source HAR file)
    - 253-hre-064.mktoresp.com
      * (more subdirectories)
    - ajax.aspnetcdn.com
      * (more subdirectories)
    - ajax.googleapis.com
      * (more subdirectories)
    - (more directories)

Each hostname has its own directory and subdirectories. This allows the user to upload all those files under the same HTTP Transaction Profile in Avalanche, and assign that Transaction Profile to one or more server IPs in their test. There are 12 hostnames in that test, so we will need to create as many Transaction Profiles in Avalanche Commander.

## Server Configuration

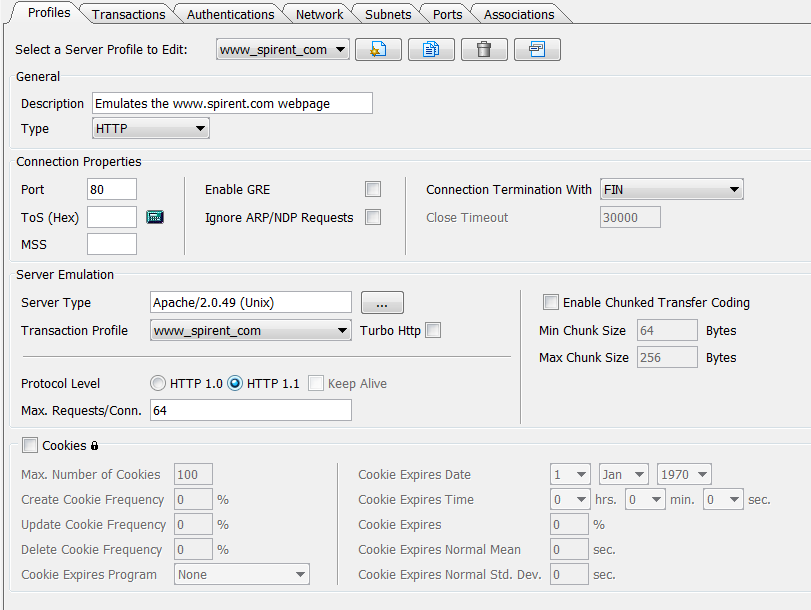
Note: This test case requires creating an Advanced Device test.

First we will create all the Transaction Profiles on the server side. Create a new profile, name it “www\_spirent\_com” and in “Body Content Type” choose “Files from directory.” Now on “Select a directory” click on the box to create a new one, then browse to where Harhar parsed the spirent.com record, and choose the “www.spirent.com” directory:

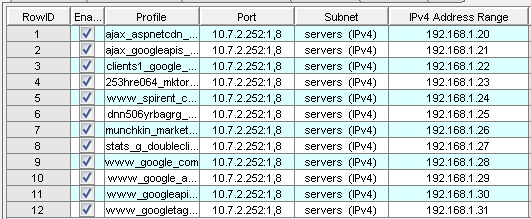


Avalanche Commander will import all the files from that directory and shouldn’t trigger any error (since Harhar sanatizes all the file names during import). Repeat this step for all the directories.

Now we need to create the Server Profiles that will return each of these Transaction Profiles. On the Server/Profiles tab, create a new server named “www\_spirent\_com” and in the “Transaction Profile” make sure you associate the right profile ; www\_spirent\_com in that case.

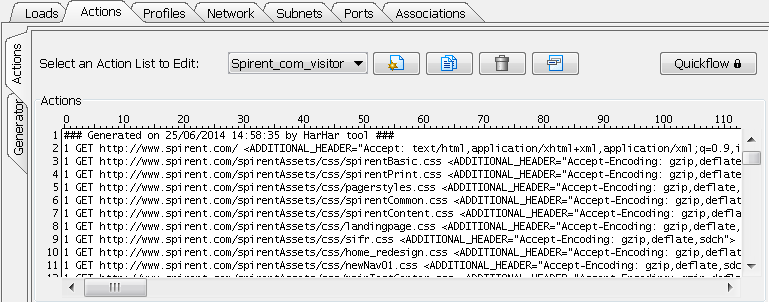


Repeat this step until each Transaction Profile is tied to a Server Profile. Once this is done, create the appropriate associations and assign fitting IP addresses to these servers. I used IP 192.168.1.20 through .31:



## Client Configuration

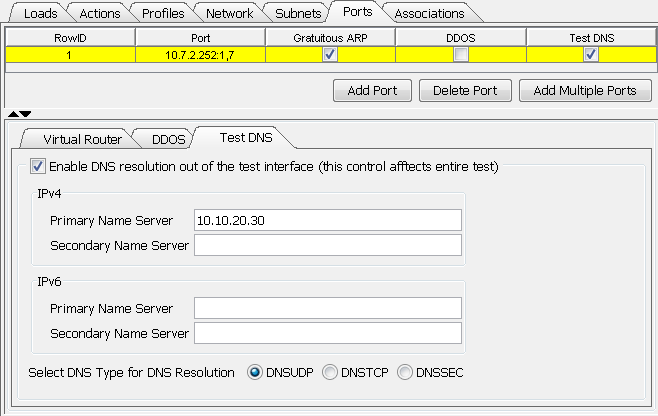
The client needs to know all the action it has to execute. Harhar generates a text file with all the actions you followed when the HAR record was created (action\_list.txt), so you simply need to copy/paste its content into an Action List. I will create a “Spirent\_com\_visitor” action list like this.



In our Action List we use hostnames in the URIs, which is realistic but requires extra configuration.

### Using a DNS Server in the Test Bed

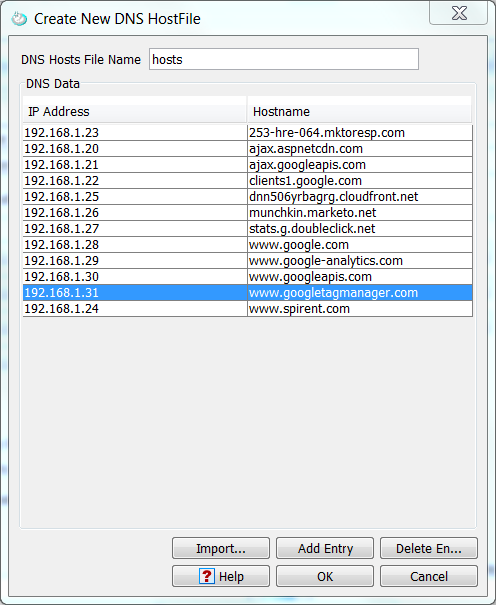
You possibly have a DNS server in your test bed, in which case you need to tell Avalanche to use it from the test port as well as specify the IP address. For instance if your DNS is on 10.10.20.30, use this configuration:



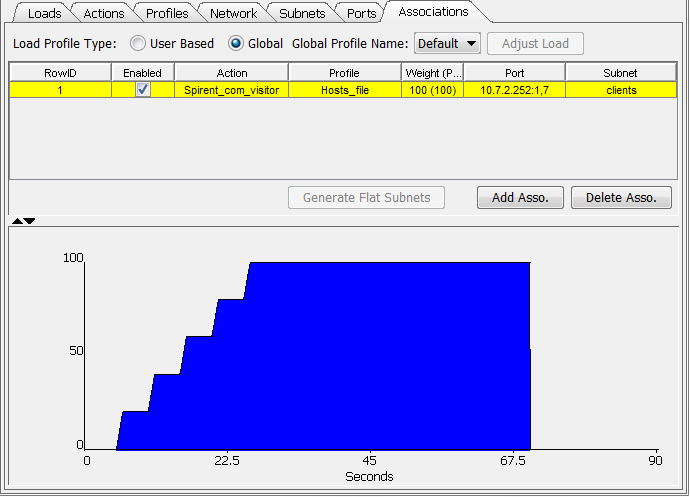
### Using a DNS Hosts File

If you don’t have a DNS server handy you can tell Avalanche which IP address corresponds to a Hostname. Note that in that case no DNS request will be sent by the clients, which may or may not be suitable. If the device you test is also in charge of handling the DNS requests in real life (a Customer Premises Equipment for instance) it’s imperative that your test case also sends these requests.

To create a DNS Host File click on Client/Profile and then on the “DNS” tab. Then check the box “Use a DNS Hosts file” and then create the file. On the new window you will be able to input the hostnames and the corresponding IPs :



To finish we’ll simply need to create a Client Association with that action list and the profile holding the DNS Hosts file, and it should all work.



# 5. Logs

Internet is made by a lot of people and companies. There are standards but they are not always followed to the letter. For instance, it happens that the “x-unknown” Media Type is sent by servers when clearly that should never happen. Because of this, some errors will sometimes be returned by Harhar. The author tried to handle as many errors as possible but it might happen that some are so weird that they weren’t.

For this reason, two log files exist:

* run.log: This file will be generated at each run if any error or warning is thrown by Harhar. Make sure to review it in case file names were truncated (for being too long) or other errors.
* UnknownMedia.log: This will be generated if a response contains a Media (MIME) Type not present in [HarNET](https://github.com/acastaner/harnet), the .NET library that Harhar uses.

# 6. License

[Spirent Communications](http://www.spirent.com/) is a leader in Test & Measurement tools. The author of this tool (Harhar) is Arnaud Castaner, an employee of Spirent. Spirent Communications was and is not involved in the building of this software and cannot be led liable for anything related to Harhar.

You can contact Spirent Communications through their website at <http://www.spirent.com> and Arnaud Castaner at [arnaud.castaner@outlook.com](mailto:arnaud.castaner@outlook.com) or [https://alarash.net/harhar/resources/linkedin-icon.png](https://www.linkedin.com/in/acastaner)

The author would like to acknowledge that this tool uses the [Newtonsoft.JSON](https://github.com/JamesNK/Newtonsoft.Json) (MIT), the [Noda Time](http://nodatime.org/) (Apache 2.0) and [HarNet](https://github.com/acastaner/harnet) (LGPL-3) libraries.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.