Krash Course - An Introduction To Krash Fuzzer

By: Andrew Brooks

Introduction

Krash is a pretty slick and straightforward block-based fuzzing app. Krash comes with the Inguma Package and is also written by the Inguma Project's head dev, Joxean Koret. While there are a lot of good tools out there for fuzzing like SPIKE or Paros Proxy, Krash is a pretty quick setup and accepts parameters of raw packets and then tokenizes them. Every token the fuzzer is able to generate will be fuzzed. This approach is similar to block based fuzzing but a bit more "automagic."

Overview

In this document, we will give a crash course in using Krash for fuzzing stateless protocols . The tools you will need are of course Inguma and all required packages (for good measure) and Wireshark. Really any tcpdump will do but I am partial to Wireshark so that is the tool of choice. We'll only be using Wireshark for grabbing some raw packets to pass to Krash so there's no need to be an expert on protocol analysis.

Let's get started!

Assuming you have downloaded the Inguma package you need to navigate to the working directory where the Krash script resides.

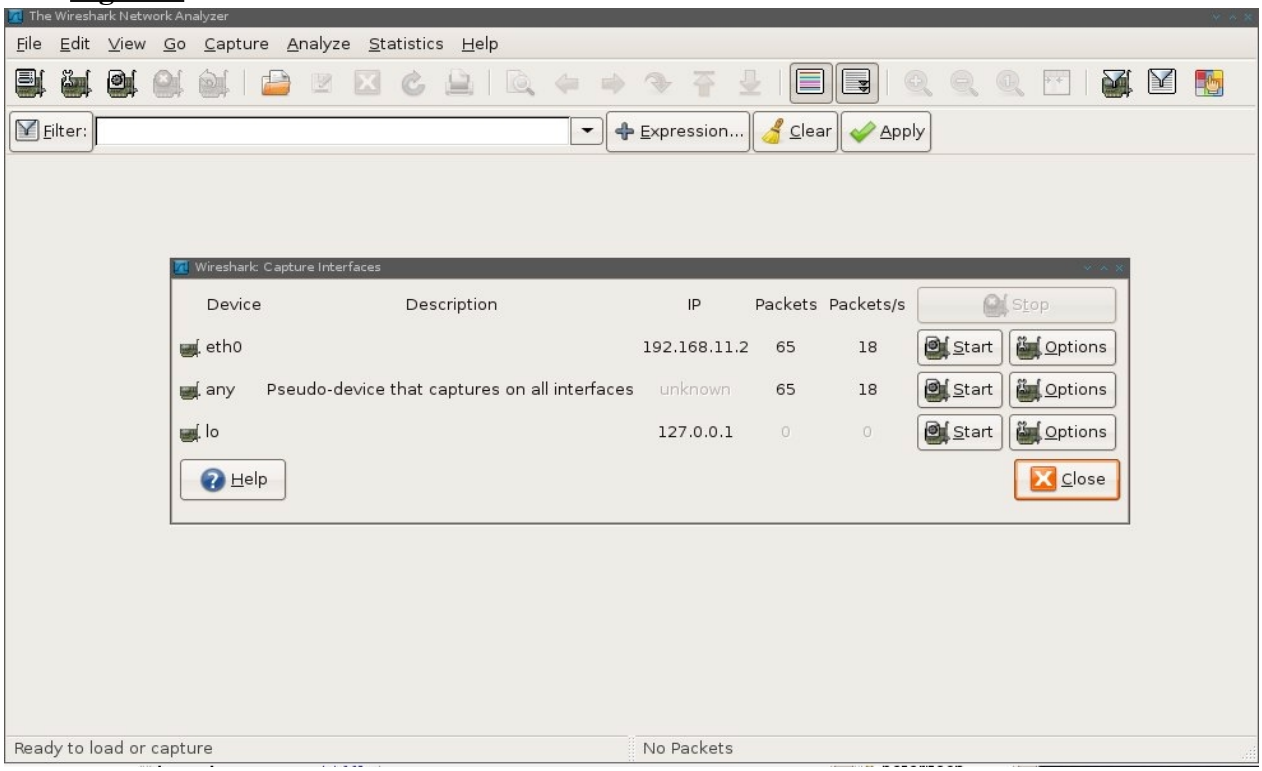
user@host:~$ cd /inguma/krash

If you navigate to the 'audits' directory within Krash, you will see folders containing supplied raw packets which can be used to fuzz just about anything you see fit. If you take a moment to read the documentation, you will notice that Krash does not currently support cookies or stateless protocols so for today we will just be fuzzing a web server, particularly HTTP traffic. We will also be using Wireshark to get a raw HTTP packet. Since this is just example, let's take whatever we find first.

First thing we need to do is open Wireshark. Since we have picked our target (thanks to Vayde for donating his server as a test subject), let's load the home page and grab the GET request for fuzzing.

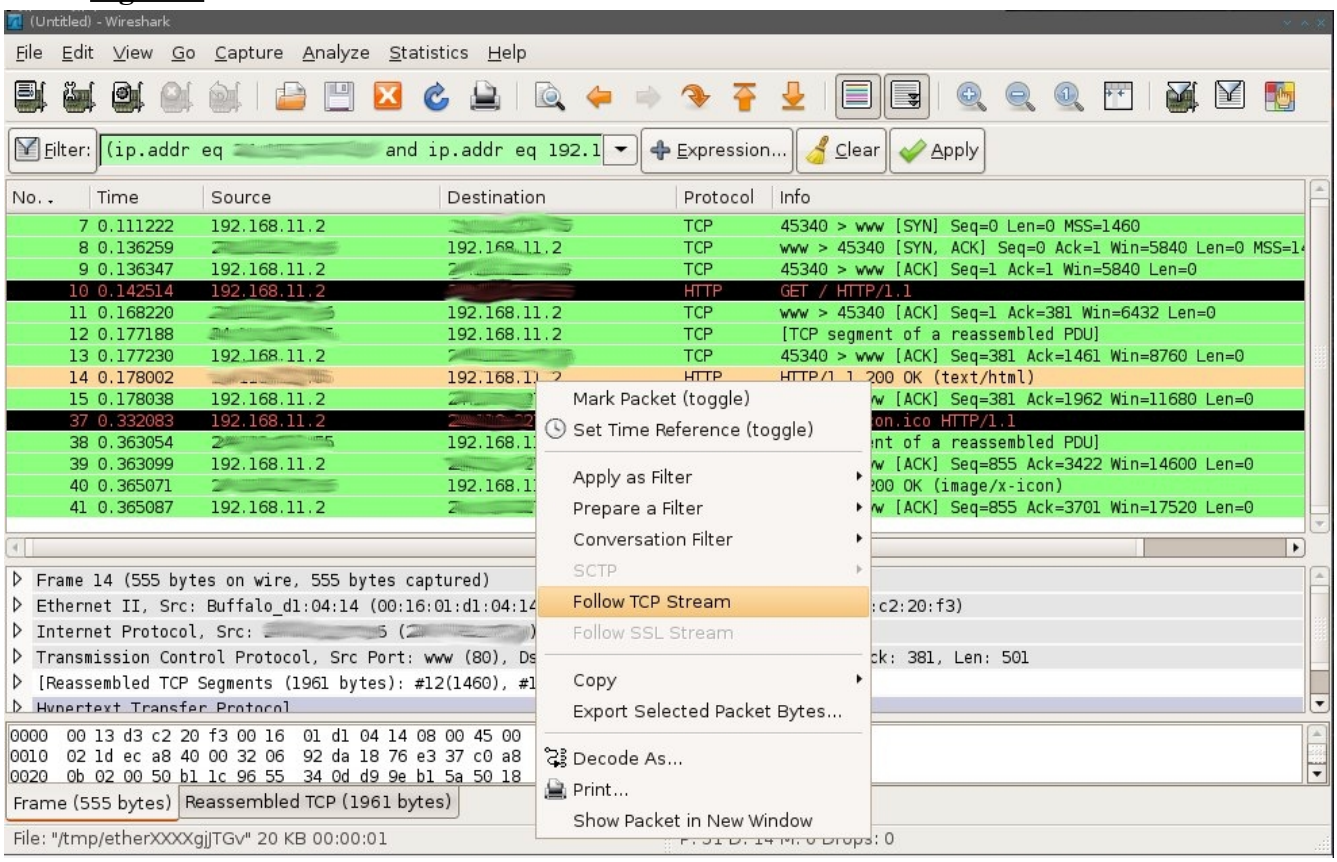
In Wireshark, start your packet capture on your listening interface. (See Figure 1)

Figure 1



After you have done that, start the packet capture and pull up the web page in any browser you'd like. Now, if you aren't familiar with Wireshark, you need to pick out the HTTP traffic, right click on the packet stream and select "follow TCP stream" from the menu. (See Figure 2)

Figure 2



Since we will be fuzzing the GET request, copy and paste that out of your window. It should look something like this:

```
GET / HTTP/1.1
User-Agent: Opera/9.50 (X11; Linux i686; U; en)
Host: CENSOREDHOST.com
Accept: text/html, application/xml;q=0.9, application/xhtml+xml,
image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1
Accept-Language: en-US,en;q=0.9
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive
```

With this securely fastened to your clipboard, open up a text processing tool of your choice and paste it into a blank document. Save the file as anything you'd like, for this example, let's just call it 'test_packet.http.' Please note that to keep things simple and organized, this should be placed in ~/inguma/krash/audits/webserver, since we are fuzzing an Apache server. Don't expect to find much fuzzing Apache though since it is widely deployed and heavily audited. Let's head back to the command line to commence fuzzing at once!

```
user@host:~$ ./krash.py audits/test_packet.http CENSOREDHOST.com 80 0
```

While most of that command is pretty easy to understand, the last 2 parameters specify the port number and verbosity respectively.

If/when you manage to find something of interest, Krash will halt and notify you of the last packet that might have caused the error in the event that you wish to recreate it.

Take care and share the knowledge!