A headless Web Browser in your Splunk Environment.

Version: 1.02 by Cédric Le Roux

Splunk, Inc.
250 Brannan Street, 2nd Floor
San Francisco, CA 94107

+1.415.568.4200(Main)
+1.415.869.3906 (Fax)
www.splunk.com

splunk>

Summary

This use case analyzes proxy logs to extract Google queries, decode it and render it in a simple statistical table. The key point of this use case is the usage of **PhantomJS WebKit** to use the Google Translation service for translation (eg: Arabic to English). Another interesting point here is the usage of arguments in lookups calls (not natively supported).

This procedure requires that you install **Faup** and **gTranslate** and has been written as a step-by-step guide for teaching purpose.

This is obvious…

Let's take some proxy logs:

```
2011-08-03 23:22:43 139 0.0.0.0 - - - OBSERVED "unavailable"
http://www.google.com/search?hl=ar&biw=1280&bih=598&tbm=isch&sa=1&q=%D8%B5%D9%88%D8%B1+%D8%B
3%D9%83%D8%B3+%D9%85%D8%AD%D8%AA%D8%B1%D9%85%D8%A9&oq=%D8%B5%D9%88%D8%B1+%D8
%B3%D9%83%D8%B3+%D9%85%D8%AD%D8%AA%D8%B1%D9%85%D8%A9&aq=f&aqi=&aql=&gs_sm=e&gs_upl=
1115614l1132856l0l1133276l23l16l0l12l12l0l598l598l5-1l1l0  200 TCP_MISS GET image/jpeg http t0.gstatic.com 80
/images ?q=tbn:ANd9GcQrDmoIhfAmFELbLPMe4y7s4lkAN3cFx8ihlMjhaNb-TYvmYnutoEFh0Bmv - "Mozilla/4.0
(compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0)" 82.137.200.47 5338 694 -
```

Do you see the user query just right here? No? … Let's render it in a more fashionable way…

Extracting URLs

First, we only keep log lines having the word "google" in it from our proxies. This example assumes there is automatic field extraction and the URLs are extracted in the "url" field.

```
sourcetype="proxy" url="*google*"
```

The resulting URL field is as follows:

```
http://www.google.com/search?hl=ar&biw=1280&bih=598&tbm=isch&sa=1&q=%D8%B5%D9%88%D8%B1+%D8%B
3%D9%83%D8%B3+%D9%85%D8%AD%D8%AA%D8%B1%D9%85%D8%A9&oq=%D8%B5%D9%88%D8%B1+%D8
%B3%D9%83%D8%B3+%D9%85%D8%AD%D8%AA%D8%B1%D9%85%D8%A9&aq=f&aqi=&aql=&gs_sm=e&gs_upl=
1115614l1132856l0l1133276l23l16l0l12l12l0l598l598l5-1l1l0
```

Keeping Google domains only

The previous search would match any log line with the word “google” in it. We could have Bing searches, Youtube queries, or any other domain, in our search results. As we only want to focus on Google queries, we use Faup to split the URLs, keeping only Google domains (whatever the TLD: google.com, google.co.uk, google.fr, etc).

```
sourcetype="proxy" url="*google*" | lookup faup url | search domain_without_tld = "google"
```

Before filtering with Faup

domain (categorical)

Appears in 90% of results
Show only events with this field

Charts
Top values by time
Top values overall

Top 10 values	#	%
google.com	43,680	76.528%
googleusercontent.com	931	1.631%
panoramio.com	480	0.841%
google.ae	426	0.746%
com.lb	286	0.501%
google.co.uk	266	0.466%

After filtering with Faup

domain (categorical)

Appears in 100% of results
Show only events with this field

Charts
Top values by time
Top values overall

Top 10 values	#	%
google.com	42,760	97.75%
google.ae	441	1.008%
google.co.uk	266	0.608%
google.de	190	0.434%
google.com.br	17	0.039%
google.ca	15	0.034%

Extracting the Query

Now, we only have Google domains and we have a field containing the whole URL. We can use the extract command to extract the queries from all those URL’s arguments.

```
sourcetype="proxy" url="*google*"| lookup faup url | search domain_without_tld = "google" | extract pairdelim="&?#", kvdelim="="
```

We now have isolated the query into the q field… but the search terms are still not obvious:

q
zara+clothes+romania
group+names
%D9%88%D8%B2%D8%A7%D8%B1%D8%A9%20%D8%A7%D9%84%D8%AA%D8%B1%D8%A8%D9%8A%D8%A9%20%D9%88%D8%A7%D9%84%D8%AA%D8%B9%D9%84%D
%D9%88%D8%B2%D8%A7%D8%B1%D8%A9%20%D8%A7%D9%84%D8%AA%D8%B1%D8%A8%D9%8A%D8%A9%20%D9%88%D8%A7%D9%84%D8%AA%D8%B9%D9%84%D
%D9%88%D8%B2%D8%A7%D8%B1%D8%A9%20%D8%A7%D9%84%D8%AA%D8%B1%D8%A8%D9%8A%D8%A9%20%D9%88%D8%A7%D9%84%D8%AA%D8%B9%D9%84%D
%D9%84%D8%A7%D8%AD%D9%88%D9%84+%D8%A5%D9%84%D8%A7+%D8%A8%D8%A7%D9%84%D9%84%D9%87
%D8%B5%D9%88%D8%B1+%D8%B3%D9%83%D8%B3+%D9%85%D8%AD%D8%AA%D8%B1%D9%85%D8%A9
%D8%A7%D9%84%D8%B9%D8%A7%D8%B2%D9%81%D8%A9%20%D9%87%D8%A8%D8%A9%20%D8%B9%D9%88%D8%AF%D8%A9

splunk>

Decoding the query

We have the queries but some of them are URLEncoded. Splunk has a native URLDecode function! The 'fields' function is used to discard all other field names unless explicitly named; we only keep the query field.

```
sourcetype="proxy" url="*google*" | lookup faup url | search domain_without_tld = "google" | extract pairdelim="&?#", kvdelim="=" | fields q | eval q=urldecode(q)
```

q
group names
zara clothes romania
العزفة هبة عودة
صور سكس محترمة
لاحول إلا بالله
وزارة التربية والتعليم
وزارة التربية والتعليم
وزارة التربية والتعليم

Oh Yeah!

We now have Google queries decoded!

This is Wonderful!

This is Amazing!

Let's make some stats on this…

… Oh wait, this is Arabic… we don't read Arabic, what does those queries means? :(

Google Translate

We can use Google Translate to decode our queries in two different ways:

- Using the Google API, but we have to pay :(
- Using the Free Web Service, but we need a web browser to render all that messy javascript.

To query Google Translate, we can forge URLs as follows:

```
http://translate.google.com/lang-in/lang-out/text
```

Where:

- **lang-in** is the language you wants to translate. If unsure, just set it to "auto" and Google will recognize the language for you.
- **lang-out** is the target language in which you want the translation to be done. For example if you want to translate French to English you will have English as lang-out.
- Finally, the **text** to translate.

Ok, the query part is easy.

Google Translate web page:

On the right, in the grey box, the translated text appears. This box is identifiable beacause of the use of the “result_box” CSS property.

So, we just have to query Google Translate and parse the result box. Easy to say… Google don’t let users to perform simple wget or similar browsing/scraping: they include tons of Javascript and often results are just not accessible “by reading the sources”. So, we need to simulate a web browser, which support Javascript.

PhantomJS

PhantomJS is a Javascript WebKit that lets you simulate a web browser that we can run from command line (see below example). From the site, “*PhantomJS is a headless WebKit scriptable with a Javascript API. It has fast and native support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG*.”

So, basically, for our needs, we need a simple javascript script which:

- Get 3 arguments (lang-in, lang-out, text to translate)
- Create and open the forged URL
- Read the result box content of the page and return it as a result

The script **gTranslate.js**, provided by **gTranslate.spl**, will use PhantomJS and JQuery to translate the text. The original version of the script was found on nousefor.net

Usage example:

```
# cd /$SPLUNK_HOME/etc/apps/gTranslate/bin
# ./phantomjs gTranslate.js auto en “Bonjour a vous !”  2>/dev/null
Hello you!
#
```

After installing the package gTranslate.spl, a new Splunk lookup is available: gTranslate. This lookup is basically a wrapper to the javascript script, which use PhantomJS go query Google Translation.

Note: this script will query Google Translate service: a network connection is required.

Lookup call with arguments

Now, we just have to call the lookup from Splunk search with arguments. In this use case, we have:

- **lang_in**: auto (let Google find for us)
- **lang_out**: en (for English)
- **text_in**: the query itself (q field)

The problem here is that **lang_in** and **lang_out** are "static" arguments, meaning they are not extracted from logs … and lookups only handle fields extracted from logs. So, we need to use the eval function to trick the lookup system.

```
.... | eval inlang="auto" | eval outlang="en" | lookup gTranslate inlang outlang intext as q
```

Let's speak Arabic!

We have our queries extracted and decoded by Splunk core functions and we have our PhantmJS lookup to query Google Translate. Cool, let's make some stats…

```
sourcetype="proxy" url="*google*" | lookup faup url | search domain_without_tld = "google" | extract pairdelim="&?#", kvdelim="=" | fields q | eval q=urldecode(q) | eval inlang="auto" | eval outlang="en" | lookup gTranslate inlang outlang intext as q | stats values(outtext) as "Translation", count by q | rename q as "Search Term" | sort -count
```

and there we go:

Overlay: None

Search Term	Translation	count
وزارة التربية والتعليم	Ministry of Education	3
group names	group names	1
zara clothes romania	zara clothes romania	1
العازفة هبة عودة	Musician gift return	1
صور سكس محترمة	Photos Sex respectable	1
لاحول إلا بالله	To turn but God	1