Word Statistics.

Version: 1.1 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 domain names using a scripted lookup named “wordstats”. The goal is to discover malicious domain names in logs (proxy, dns, etc). This is an attempt to highlights weird domains. In addition to wordstats, the faup lookup will be used to “sanitize” domain names (removing the TLD of a domain name for example).

This document has been written as a step-by-step guide.

Faup

This use case requires Faup to parse URLs but this document does not cover its installation. Please refer to the official documentation.

Once you have installed Faup, you should be able to use it like in the following example where the used field is “cs_uri” and its renamed to “url” for Faup:

```
.... | lookup faup url as cs_uri
```

New fields will be created like domain, domain_without_tld, tld, etc.

WordStats

WordStats is a scripted lookup installed by the **wordstats.spl** apps (available on SplunkBase), which count things. Basically, the first idea is to count vowels, consonants, digits, dashes, dots, Shannon’s entropy, etc, of submitted words in order to identify “weird” words. The command by itself does not embed intelligence.

In this use case, words will be domain names and the goal is to identify weird domains like malwares C&C.

To call wordstats, the following generic command must be used:

```
... | lookup wordstats field as the_name_of_your_field
```

The following example uses the field **cs_host** from proxy logs as source for Faup, which return the **domain_without_tld** field (ex: google). The **dedup** command is used to only send unique domains names to the **wordstats** command.

```
index=proxy | lookup faup url as cs_host | dedup domain_without_tld | lookup wordstats field as domain_without_tld
```

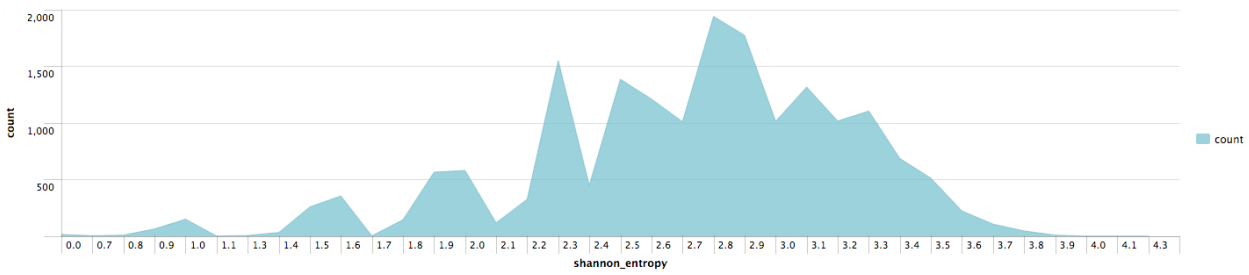
domain_without_tld	ws_entropy	ws_length	ws_nbConsonant	ws_nbDash	ws_nbDigit	ws_nbDot	ws_nbElse	ws_nbPunctuation	ws_nbUnderscore	ws_nbVowel	ws_usePunnyCode
top-sy	2.584962500721156	6	3	1	0	0	0	1	0	2	0
assafir	2.2359263506290326	7	4	0	0	0	0	0	0	3	0
facebook	2.75	8	4	0	0	0	0	0	0	4	0
ishtartv	2.75	8	6	0	0	0	0	0	0	2	0
94.76.197.100	2.7192945256669794	13	0	0	10	3	0	3	0	0	0
stagevu	2.8073549220576046	7	4	0	0	0	0	0	0	3	0
desert-operations	3.3371753411230776	17	9	1	0	0	0	1	0	7	0
msn	1.584962500721156	3	3	0	0	0	0	0	0	0	0
worldpress	2.9219280948873627	10	8	0	0	0	0	0	0	2	0
roomarab	2.25	8	4	0	0	0	0	0	0	4	0
gstatic	2.5216406363433186	7	5	0	0	0	0	0	0	2	0
bing	2.0	4	3	0	0	0	0	0	0	1	0
mail	2.0	4	2	0	0	0	0	0	0	2	0
purenetworks	3.2516291673878226	12	8	0	0	0	0	0	0	4	0
ceipmsn	2.8073549220576046	7	5	0	0	0	0	0	0	2	0
yahoo	1.9219280948873623	5	1	0	0	0	0	0	0	4	0

Note that all fields created by wordstats start with “ws_”:

- **ws_entropy** : shannon’s entropy of the word
- **ws_length**: length of the word.
- **ws_nbConsonant**: number of consonants in the word (upper/lower case independent)
- **ws_nbVowel**: number of vowels in the word (upper/lower case independent)
- **ws_nbUppercase**: number of characters that are uppercase
- **ws_nbLowercase**: number of characters that are lowercase
- **ws_nbDash**: number of dashes in the word
- **ws_nbNumeric**: number of digits in the word
- **ws_nbDot**: number of dots in the word
- **ws_nbUnderscore**: number of underscores in the word
- **ws_usePunnyCode**: whether the word is Punnycode or not (start with the pattern “xn--“)
- **ws_isIPv4**: whether the word is IPv4 or not (regex is too permissive so 99.9.999.9 will match)
- **ws_containsIPv4**: whether the word contains an IPv4 pattern or not (match foo99.99.99.99bar for example).
- **ws_nbPunctuation**: number of characters belonging to the following set: ['\'', '"', '[',']','(',')','{','}','<','>',':',',','-','!','.', '?',';','/'] (please see http://en.wikipedia.org/wiki/Punctuation)
- **ws_nbTypography**: number of characters belonging to the following set: ['&', '*', '@', '\\', '^', '#', '%', '~', '_', '|'] (please see http://en.wikipedia.org/wiki/Punctuation)
- **ws_nbElse**: number of characters that were not counted elsewhere (like the equal sign for example)
- **ws_nbAlpha**: a shortcut for the sum of **ws_nbConsonant** and **ws_nbVowel**
- **ws_nbAlphaNum**: a shortcut for the sum of **ws_nbAlpha** and **ws_nbNumeric**

Now, the shannon entropy will be rounded and a simple statistical chart will be realized to analyze the distribution of domain’s entropy. In our sample data (around 7 million of proxy log lines), the following graph has been obtained using the query:

```
index=proxy cs_host != "" | lookup faup url as cs_host | dedup domain_without_tld | lookup wordstats field as domain_without_tld | eval shannon_entropy=round(ws_entropy, 1) | stats count by shannon_entropy
```



Obviously, in our context, very few domains have entropy larger than 3.5 or smaller than 1.4. Let's have a look to them:

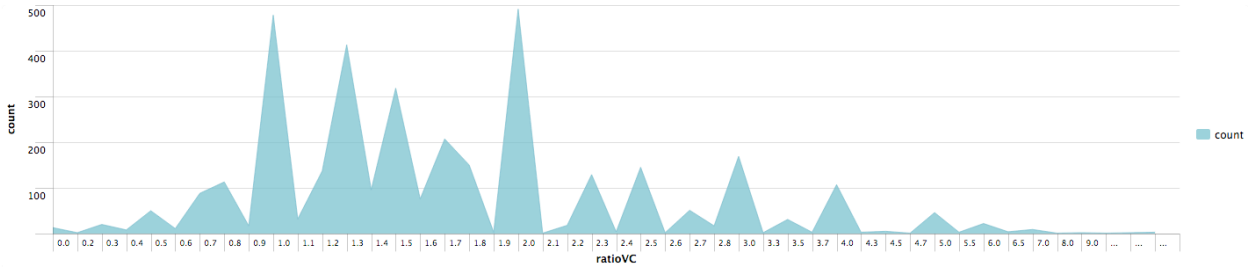
```
index=proxy cs_host != ""| lookup faup url as cs_host | dedup domain_without_tld | lookup wordstats field as domain_without_tld | eval shannon_entropy=round(ws_entropy, 1) | search (shannon_entropy > 3.5) OR (shannon_entropy < 1.4)
```

domain_without_tld	ws_entropy	ws_length	ws_nbConsonant	ws_nbDash	ws_nbDigit
0zz0	1.0	4	2	0	2
wwe	0.9182958340544896	3	2	0	0
platinum-tube-porn	3.614369445886757	18	10	2	0
checkyourlibidonow	3.6835423624332306	18	10	0	0
9q9q	1.0	4	2	0	2
xnxx	0.8112781244591328	4	4	0	0
g9g	0.9182958340544896	3	2	0	1
cz	1.0	2	2	0	0
-	0.0	1	0	1	0
64.4.44.44	1.295461844238322	10	0	0	7
socialsexnetwork	3.625	16	10	0	0
qq	0.0	2	2	0	0
vi	1.0	2	1	0	0
sns	0.9182958340544896	3	3	0	0
loading-resource	3.625	16	8	1	0
upsbkschmajhlxs6	3.577819531114783	16	13	0	1
vk	1.0	2	2	0	0
xlxx	0.8112781244591328	4	4	0	0
s3	1.0	2	1	0	1
9kk9	1.0	4	2	0	2

A quick view to those results allow us to identify the really weird domain "ubtvurgccjklqpxh.net" which is in fact known to having hosted malwares contents (verified with a simple google query).

With the same principle, we can for example compute the ratio of the number of vowels versus the number of consonants in each word.

```
index=proxy cs_host != "" | lookup faup url as cs_host | dedup domain_without_tld | lookup wordstats field as domain_without_tld | eval ratioVC=round(ws_nbConsonant/ws_nbVowel, 1) | stats count by ratioVC
```



Again, we see an "average zone" where all ratios are located, but we see also peak values which might be interesting to identify malware C&C. This behavior reflects the fact that "normal" domains names are supposedly reflecting words, mostly English, and some malwares domains names are using Domain Generation Algorithm (pseudo randomness).

Again, we can spot some weird domains that might or might not correlate with shannon entropy results (see screenshot).

```
index=proxy cs_host != "" | lookup faup url as cs_host | dedup domain_without_tld | lookup wordstats field as domain_without_tld | eval ratioVC=round(ws_nbConsonant/ws_nbVowel, 1) | search ratioVC > 4.0 OR ratioVC < 0.6 | sort –ratioVC
```

domain_without_tld	ratioVC	ws_entropy	ws_length	ws_nbConsonant
shkoznssmwrblfqx	15.0	3.702819531114783	16	15
fplqsxitksmqgpwf	15.0	3.5	16	15
mdmhvvvzrrrzwhko	15.0	3.0306390622295662	16	15
uqdvmrjknqqqxrsq	15.0	3.1493974703476995	16	15
ufvhkcnwjwphrqxm	15.0	3.75	16	15
ovmvlnqlnmvssnkt	15.0	3.0306390622295662	16	15
tspwlulsxnkvwnq	14.0	3.3735572622751855	15	14
sxxrwlvqnpxplxu	14.0	3.106890595608519	15	14
wtgspshjbjgiqqz	14.0	3.3735572622751855	15	14
pmnjqkhkfqfmonp	14.0	3.1068905956085184	15	14
bgjokttxtpslkp	13.0	3.182005814760214	14	13
ztjgqjirrlwwtj	13.0	3.039148671903071	14	13
phptusttqjmn	11.0	3.0220552088742	12	11

As a conclusion, this lookup, wordstats, is just a tool to help analysts to search for weird domains for example but by itself, this command has no “intelligence”. The same conclusion can be applied to the Shannon entropy, which can be used as an indicator, but not as a “malware revealer” or an “indicator of compromise” by itself.