Paul 2 - determine...

```
%pyspark                                                                    FINISHED

# Zeppelin notebook to deterine coverage of CC domains against the DNS-BH Malware
# Domain Blocklist from this site: http://mirror1.malwaredomains.com
# Specifically, the 'justdomains' file, which currently contains 31k 'bad' domains
# PJ - 21 Sept 2017

import boto
from pyspark.sql.types import *

# Import the DNS-BH domain list as a DataFrame
bh_schema=StructType([StructField("domain", StringType(), False)])
dns_bh=spark.read.csv('s3://billsdata.net/CommonCrawl/DNS-BH/justdomains.dms', header=F
dns_bh.show(3)
print("Bad domains in DNS-BH: " + str(dns_bh.count()))
dns_bh.cache()
```

```
+--------------------+
|              domain|
+--------------------+
|amazon.co.uk.secu...|
|autosegurancabras...|
|christianmensfell...|
+--------------------+
only showing top 3 rows
Bad domains in DNS-BH:31877
DataFrame[domain: string]
```

```
%pyspark                                                                    FINISHED

# Load a list of CC domains, using the host-level web graph produced by CommonSearch in
# https://about.commonsearch.org/2016/07/our-first-public-datasets-host-level-webgraph-
# TO DO: Maybe we can derive a better list than this ourselves?

schema=StructType([StructField("id", LongType(), False), StructField("domain", StringTy
#cc_domains=spark.read.csv('s3://billsdata.net/CommonCrawl/DNS-BH/vertices.txt.gz', hea
cc_domains=spark.read.option("header", "false").option("inferSchema", "true").option("d
#cc_domains=spark.createDataFrame(cc_domains,schema) # apply schema
cc_domains.show(3)
```

```
+-------------------+--------------------+
|                 id|              domain|
+-------------------+--------------------+
|8549019527203575649|widgets.lumberjoc...|
|         -819422369|        babybjorn.it|
|         1252628295|      albergoelea.it|
+-------------------+--------------------+
only showing top 3 rows
```

```
%pyspark
```

FINISHED

```
# Remove www prefixes and perform other clean-up on both lists
from pyspark.sql.functions import UserDefinedFunction
from pyspark.sql.types import StringType

name='domain'
prefix="www."
udf = UserDefinedFunction(lambda x: (x[len(prefix):] if (x.startswith(prefix) if x else

dns_bh2 = dns_bh.select(*[udf(column).alias(name) if column == name else column for col
cc_domains2 = cc_domains.select(*[udf(column).alias(name) if column == name else column
dns_bh2.show(3)
cc_domains2.show(3)
```

```
+--------------------+
|              domain|
+--------------------+
|amazon.co.uk.secu...|
|autoseguranca bras...|
|christianmensfell...|
+--------------------+
only showing top 3 rows
+-------------------+--------------------+
|                 id|              domain|
+-------------------+--------------------+
|8549019527203575649|widgets.lumberjoc...|
|         -819422369|        babybjorn.it|
|         1252628295|      albergoelea.it|
+-------------------+--------------------+
only showing top 3 rows
```

```
%pyspark
```

FINISHED

```
# Join the two lists of domains, and compute overlap
common_domains=dns_bh.join(cc_domains, ["domain"])
common_domains.show(3)
print("Number of common domains: " + str(common_domains.count()))

# So we have 3905 common domains (using full domain string)
# Not great, but probably enough to start training a predictive model -  see 'Paul 3 -
# TODO: For some reason, the www. filtering in cell above causing Join not to complete
```

```
+--------------------+-----------+
|              domain|         id|
+--------------------+-----------+
|    aeroclubparma.it|-1573485380|
|higherpositions.info| 1149031776|
|    artlegendsoc.org| 1509011612|
+--------------------+-----------+
only showing top 3 rows
Number of common domains: 3905
```

```
%pyspark
```

READY