Chapter 2: Learning The “Hello World” Of Security Data Analysis

“from one thing, know ten thousand things”

― Miyamoto Musashi, The Book of Five Rings: Miyamoto Musashi

If you’ve ever tried to learn a new programming language there’s a good chance you started of with a “Hello World” example that quickly introduces basic language structure and code execution. The immediate sense of accomplishment as the syntax is verified by the compiler/interpreter and the familiar two-word output is displayed becomes a catalyst for the notion that, soon, you shall have the ability to bend this new language to your will.

This chapter takes the “Hello World” concept and expands it to a walk-through of a self-contained, introductory security data analysis use case that you will be able to follow along, execute and take concepts from as you start to perform your own analyses. There are different examples for various types of analyses in Python and R to give you an idea of the similarities, strengths and differences between both languages in a real life example context. If you’re not familiar with one or both of those languages you should read Chapter 1.5 first and at least skim some of the external resources referenced there. Remember, all the source code, sample data and visualizations are on the book’s web site, so no need for transcription, just focus on the flow of the analyses.

Preparing For Analysis

Before jumping into data retrieval and analysis, we need to setup an area where we can keep input data, analysis scripts, output (visualizations, reports and/or data) and any supporting documentation organized. For the purposes of this chapter, we’ll be using the following directory structure:

/book/chapter3/reputation

|-R

|-data

|-docs

|-output

|-python

|-support

|-tmp

Like most elements of programming, there is no one, true way to setup this structure for analyses, but you should strive to find one that works for you and stick with it. A great way to do that is to take a lesson from modern web framework builders and use a simple setup script that builds the structure for you:

Sample analysis preparation script

#!/bin/sh

#

# prep: prep analytics directory structure

#

# usage: prep DIRNAME

#

DIR=$1

if [ ! -d "${DIR}" ]; then

mkdir -p ${DIR}/R \

${DIR}/data \

${DIR}/docs \

${DIR}/output \

${DIR}/python \

${DIR}/support \

${DIR}/tmp

> ${DIR}/readme.md

ls -lR ${DIR}

else

echo "Directory "${DIR}" already exists"

fi

Once the structure is in place, it’s time to retrieve, explore and analyze some data.

Getting Data

We are living in the golden age of data in information security. The challenge is no longer where to get data from, but what to do with it. Figure 3-1 lists many of the common internal and external sources and types of data, and—as you’ll see in the rest of the book—the kind of information in each will drive the type of research you perform.

For this use case, we’ll be working with AlienVault’s IP Reputation Database (<http://labs.alienvault.com/labs/index.php/projects/open-source-ip-reputation-portal/download-ip-reputation-database/>), a free data set that contains information on various types of “badness” across the internet. AlienVault provides this data in numerous formats and the version we’ll be working with is the OSSIM Format (<http://reputation.alienvault.com/reputation.data>) as it provides the richest information of the ones available.

type="tip"

AlienVault updates their IP reputation data set hourly and produces a companion “revision” file (<http://reputation.alienvault.com/reputation.rev>), enabling you to ensure you are working with the latest data set or keep a history of data sets.

When performing a one-off, exploratory analysis or getting a first look at a data set, it’s acceptable to just do a quick download via browser. If we do that for AlienVault IP reputation database and take a look at the first few data elements we can get an idea of the contents and format, which will come in handy when we start to read in and work with the data.

Performing a quick review of the downloaded data set

$ **head -10 reputation.data** *# look at the first few lines in the file*

222.76.212.189#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

222.76.212.185#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

222.76.212.186#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

5.34.246.67#6#3#Spamming#US##38.0,-97.0#12

178.94.97.176#4#5#Scanning Host#UA#Merefa#49.823001861,36.0507011414#11

66.2.49.232#4#2#Scanning Host#US#Union City#37.59629821,-122.0656966#11

222.76.212.173#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

222.76.212.172#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

222.76.212.171#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

174.142.46.19#6#3#Spamming###24.4797992706,118.08190155#12

$ **wc –l reputation.data** *# see how many total records there are*

258626 reputation.data

For most projects it’s better to get into the habit of retrieving the data source directly from your analysis scripts. If you still prefer to download files manually you should provide some type of comment in your analysis scripts that documents where the source data comes from and when you retrieved the data to make it easier to repeat the analyses at a later date.

The following snippets show how to perform the data retrieval in both R and Python. If you are following along with RStudio or iPython, the both the R and Python code snippets assume a working directory of the top level of the project structure (e.g. executing from “reputation” directory.)

R code to download the AlienVault data

# URL for the AlienVault IP Reputation Database (OSSIM format)

# storing the URL in a variable makes it easier to modify later

# if it changes

avURL <- "http://reputation.alienvault.com/reputation.data"

# relative path for the downloaded data

avRep <- "data/reputation.data"

# using an if{}-wrapped test with download.file() vs read.xxx() avoids

# having to re-download a 16MB file every time we run the script

if (file.access(avRep)) {

download.file(avURL,avRep)

}

Python code to download the AlienVault data

#!/usr/bin/python

#

# reputation.py

#

# sample analysis script for AlienVault IP Reputation Database data

#

# URL for the AlienVault IP Reputation Database (OSSIM format)

# storing the URL in a variable makes it easier to modify later

# if it changes

import urllib

import os.path

avURL = "http://reputation.alienvault.com/reputation.data"

# relative path for the downloaded data

avRep = "data/reputation.data"

# using an if-wrapped test with urllib.urlretrieve() vs direct read

# via panads avoids having to re-download a 16MB file every time we

# run the script

if not os.path.isfile(avRep):

urllib.urlretrieve(avURL, filename=avRep)

The R and Python code look very similar and follow the same basic structure using variables whenever possible for URL and filenames and testing for the existence of the data file before downloading it again. With the IP reputation data in hand, it’s now time to read in the data so we can begin to work with it.

Reading In Data

Both R and Python (with pandas) abstract quite a bit of complexity when it comes to reading and parsing data into structures for processing. R’s read.table(), read.csv(), read.delim() and pandas read\_csv() will cover nearly all your delimited file reading needs and provide robust configuration options for even the most gnarly input file. Both tools, as we’ll see in later chapters, provide ways to retrieve data from SQL and “NoSQL” databases, HDFS “big data” setups and even process unstructured data.

From our cursory examination of the downloaded file, we can see the AlienVault data has a fairly straightforward record format with eight primary fields using a “#” as the field separator.

222.76.212.189#4#2#Scanning Host#CN#Xiamen#24.479799270,118.08190155#11

The consistency in the record format makes the consumption of the data equally as straightforward in each language.

R code to read in the AlienVault data

# read in the IP reputation db into a data frame

av <- read.csv(avRep,sep="#",stringsAsFactors=FALSE)

# take a quick look at the data structure

head(av)

colnames(av) <- c("IP","Reliability","Risk",

"Type","Country","Locale","Coords","x")

Python code to read in the AlienVault data

import pandas as pd

# read in the data into a pandas data frame

av = pd.read\_csv(avRep,sep="#")

# take a quick look at the data structure

print(av)

av.columns = ["IP","Reliability","Risk","Type","Country",

"Locale","Coords","x"]

Since the reputation data file lacks a header, the code above assigns more meaningful column names manually. This is a completely optional step, but it will help avoid confusion as you expand your analyses and, as we’ll see further in this chapter, help build consistency across data frames if you bring in additional data sets.

Exploring Data

It’s now time to bring your security domain expertise into the discussion as we explore this data set to see what is interesting and help us form good questions to ask and answer. Despite this data set having almost 260,000 records, we have many tools at our disposal to help get a feel for what it contains.

There are some tidbits of information we know about the data even before we take a programmatic look:

* we know each record is associated with a unique IP address, so there are 258,626 IP addresses
* we know some attempt has been made to discern how reliable the IP address classification is
* we know some attempt has been made to discern the level of “risk” associated with each IP address
* we know each IP address has been pre-geo-located for us

With this knowledge, we can first ascertain that there is little efficacy in looking at the IP data in raw form since that is the equivalent of the primary key for this data set. However, we can use the fact that the “Reliability”, “Risk”, “Type” and “Country” fields are the equivalent of categorical data: i.e. they enable dividing the data set into groups. We can use the summary() and factor() functions in R to see counts of these groupings.

**summary(factor(av$Reliability))**

1 2 3 4 5 6 7 8 9 10

5612 149117 10892 87039 7 4758 297 21 686 196

**summary(factor(av$Risk))**

1 2 3 4 5 6 7

39 213851 33719 9588 1328 90 10

**summary(factor(av$Type))**

APT;Malware Domain

1

C&C

610

C&C;Malware Domain

31

C&C;Malware IP

20

C&C;Scanning Host

7

Malicious Host

3770

Malicious Host;Malware Domain

4

Malicious Host;Malware IP

2

Malicious Host;Scanning Host

163

Malware distribution

1

Malware distribution;Malicious Host

1

Malware distribution;Malware IP

4

Malware Domain

9274

Malware Domain;C&C

25

Malware Domain;Malicious Host

4

Malware Domain;Malware IP

173

Malware Domain;Scanning Host

39

Malware Domain;Spamming

2

Malware IP

6470

Malware IP;C&C

2

Malware IP;Malicious Host

1

Malware IP;Malware Domain

57

Malware IP;Scanning Host

8

Malware IP;Spamming

7

Scanning Host

234179

Scanning Host;C&C

2

Scanning Host;Malicious Host

215

Scanning Host;Malware Domain

19

Scanning Host;Malware IP

7

Scanning Host;Spamming

7

Spamming

3487

Spamming;Malware Domain

5

Spamming;Malware IP

4

Spamming;Scanning Host

24

**summary(factor(av$Country))**

CN US TR DE NL RU GB

68582 50387 13958 10055 9953 7931 6346 6293

IN FR TW BR UA RO KR CA

5480 5449 4399 3811 3443 3274 3101 3051

AR MX TH IT HK ES CL AE

3046 3039 2572 2448 2361 1929 1896 1827

JP HU PL VE EG ID RS PK

1811 1636 1610 1589 1452 1378 1323 1309

VN LV NO CZ BG SG IR IL

1203 1056 958 928 871 868 866 854

PT BE MD MY SA ZA GR PA

847 834 788 664 582 573 557 554

PH BD LB IS UY CH KZ CY

552 535 517 516 516 333 313 295

PE FI LU EC EE NZ KW A1

295 286 283 278 274 272 269 267

GT KH DO PY AW AO NI IE

261 261 259 259 257 256 256 201

AU SE EU LT VG DK AT BY

155 130 129 65 59 54 51 35

GE CO SK HR PS SI JO BA

34 33 31 25 23 20 16 15

MK MA BO TN BZ AZ MN PR

14 13 10 10 8 7 7 7

AM LK MQ (Other)

6 5 5 105

The numerical tables help, but a picture can provide a whole new perspective on the data set.

Figure 3-2 Bar Charts of Reliability, Risk, Type and Country Factors [f0302.eps]

Asking A Question

Augmenting Data