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 side-by-side examples for both 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 it references. 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

#

# 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 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**

222.76.212.189#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

222.76.212.185#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

222.76.212.186#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

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

178.94.97.176#4#5#Scanning Host#UA#Merefa#49.8230018616,36.0507011414#11

66.2.49.232#4#2#Scanning Host#US#Union City#37.5962982178,-122.065696716#11

222.76.212.173#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

222.76.212.172#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

222.76.212.171#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

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

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 listings show how to do the data retrieval in both R and Python.

R code to download the AlienVault data

If running this by hand, the code assumes the directory structure described above and the R working directory set to the top level directory

# 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

If running this by hand, the code assumes the directory structure described above and script execution from the top level directory

#!/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 using a “#” as field separator.

222.76.212.189#4#2#Scanning Host#CN#Xiamen#24.4797992706,118.08190155#11

This 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="#")

# take a quick look at the data

head(av)

# assign more readable column names to make it easier to work with the data

# IP | reliability | risk | type | country | locale | coords | x

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 dat

print(av)

# assign more readable column names to make it easier to work with the data

# IP | reliability | risk | type | country | locale | coords | x

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 sets.

Exploring Data

Asking A Question

Augmenting Data