Incident Response Programming with R

Eric Zielinski
Sr. Consultant, Nationwide
About Me?

• Cyber Defender for Nationwide
• Over 15 years in Information Security
• Speaker at various conferences FIRST, CEIC, FS-ISAC etc.
• Focus on blue team activities such as Forensics, Incident Response, and Data Exfiltration
• 4\textsuperscript{th} most punctual guy I know
Agenda

- Why R?
- Overview of R
- Reading data sets
- Case Study
- Extending R with packages
Disclaimer

This presentation will not teach you how to become an expert programmer in R in under 45min
So What Will This Teach Me?

• How we can use data analytics to speed up our response and for post lessons learned
• How we should leverage programming languages more often in incident response
• How we can develop our own tools and analytics
• This is not trying to replace your current practices. Just simply giving you another tool in your toolbox, it’s really up to you on how you use it.
Frequently Asked Questions
Issues...

- Incident response has been very *nix focused for years. This is not a bad thing, *nix rocks!
- The problem is that we are just not that good at detecting incidents
- So how can we get better?
  - Do we need to speed up response times?
  - Do we need better tools?
  - Do we need better talent?
  - Do we need more skills?
- So for IR there must be a different way, right?
- We must change our ways of thinking and try something new!
The good news

• Often times we are dealing with the same data sets
  • We see a lot of the same log files, config files, data sets, etc…
  • Shouldn’t we be able to streamline these?

• What if we take more time to understand the data so future responses can be faster!
  • Think post incident work!!
Incident Response

Neanderthal method

Bang on keyboard and mouse until you find something

Sophisticated Neanderthal Method

Let the data work for you, organize your data, combine, analyze, and respond
Lifecycle

- Preparation
- Detection
- Analysis
- Containment
- Post Incident Activity
Post Incident Evolution

• Analysts often spend over 80% of their time preparing and exploring data sets before they begin more formal analysis work
Why R?

• R runs quickly
• It’s intuitive
• Vectorized programming
• It’s interactive!
  • View(Logs)
Quirks

Nerd Quirk #1

Knowing the difference between an acronym and an initialism.
Where to begin?

• Installing R

• R Studio vs R Project
  R Project - http://cran.r-project.org/
  R Studio - http://www.rstudio.com/

• There are thousands of packages!
R Project

R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

   Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()', 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.65 (6833) x86_64-apple-darwin10.8.0]

[Workspace restored from /Users/zieline/Desktop/R/.RData]
[History restored from /Users/zieline/Desktop/R/.Rapp.history]

> |
RStudio
Memory

• How much memory is required to store data set in memory?

• How many rows and columns does your dataset contain?
  • 1,500,000 rows & 120 columns (all numeric data)
  • each number requires 8 bytes of memory
  • numbers are stored using 64 bit numbers
  • 8 bits per byte, so 8 bytes of memory per numeric object
    • 1,500,000x120x8 bytes/numeric
    • 144000000 bytes
    • 1373.29 MB
    • 1.34 GB Memory required.
  • Need a lil more than this to run, but not much more.
Up and Running

• Set your path for R to read your data sets from
• Installing packages (thousands of packages)
• Swirl - http://swirlstats.com/
• Lets see some commands!
Overview of R

- Syntax example (storing numbers)
  - X <- c(10.4, 5.6, 2.3, 4.5 or whatever)

```
> x <- c(1,2,3,4,5,6,7,8,9,10)
> x
[1]  1  2  3  4  5  6  7  8  9 10
```

- Syntax example (storing strings)
  - X <- “string”

```
> x <- "string"
> x
[1] "string"
```
Quick Overview of R

• Data Types
• Objects
• Control structures – uses standard control structures
  • If else
  • For
  • While
  • Switch
• Functions
  • Fundamental building blocks of R
  • Functions are objects
  • 3 main objectives
    • Body ()
    • Formals ()
    • Environment ()
Getting started on reading Data

- Multiple ways to read data into R
  - Read.table, read.csv
  - readLines
  - Source
  - Dget
  - Load
  - Unserialize
Reading Data

Import an entire log file into a variable

data <- read.table("logfile.txt")

• File – where to get the data
• Header – indicates header line
• Sep – how columns are separated
• StringsAsFactors
• colNames – Names of the columns
Connections

- File – opens connection to file
- Gzfile – opens connection to gzip
- Bzfile – opens connection to bzip2
- url – opens connection to webpage
Cleaning up the memory mess

Your friends:

`rm(list=ls())` – removes everything from memory

`ctrl + L` – clears the console
Now it’s time to dance!
Case Study

Web logs
Step 1: Gather the logs

You don't have to see the whole staircase, just take the first step.
Step 2: Parse the logs

Ambition is the first step to success. The second step is action.
Step 3: Analyze the data in R

Reason's last step is the recognition that there are an infinite number of things which are beyond it.

(Blaise Pascal)
Case Study: Reading the data

apachelogs <- read.csv(
  file = "other_vhosts_access.log"
  , sep = " "
  , header = FALSE
  , stringsAsFactors=FALSE)

Example of log files

- Apache weblogs without column names

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
<th>V7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET / HTTP/1.1</td>
<td>GET /css/base.css HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /css/ui.tabs.css HTTP/1.1</td>
<td>GET /js/ui.core.js HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>127.0.1.1:80</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:16:14:30] +0000] GET /images/webmin.png HTTP/1.1</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
<td></td>
</tr>
</tbody>
</table>
Diving deeper

• Understand your log format

• Apache log format

127.0.1.1:443 192.168.72.1 - - [17/May/2015:17:41:02 +0000] "GET /images/cage.png HTTP/1.1" 200 4792
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.152
Safari/537.36"
Case Study: Reading the data

apachelogs <- read.csv(
    file = "other_vhosts_access.log",
    sep = " ",
    header = FALSE,
    stringsAsFactors=FALSE,
    col.names = c("Remote Host","Destination Host", "NULL1", "NULL2", "Date", "Zone", "Url Request", "Response Code", "Bytes", "Response", "User Agent"))
Example of log files

- Apache weblogs without column names

<table>
<thead>
<tr>
<th>Remote.Host</th>
<th>Destination.Host</th>
<th>NULL1</th>
<th>NULL2</th>
<th>Date</th>
<th>Zone</th>
<th>UrlRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /css/base.css HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /css/ui.tabs.css HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /js/ui.core.js HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /js/ui.tabs.js HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /images/webmin.png HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /images/phpmyadmin.png HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /js/jquery-1.2.6.js HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /images/tab.png HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /images/shell.png HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET /favicon.ico HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:10:14:30]</td>
<td>+0000</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:18]</td>
<td>+0000</td>
<td>GET /database-offline.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:18]</td>
<td>+0000</td>
<td>GET /styles/global-styles.css HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:18]</td>
<td>+0000</td>
<td>GET /favicon.ico HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:32]</td>
<td>+0000</td>
<td>GET /set-up-database.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:38]</td>
<td>+0000</td>
<td>GET /index.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:56]</td>
<td>+0000</td>
<td>GET /index.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:18:56]</td>
<td>+0000</td>
<td>GET /database-offline.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>*</td>
<td>*</td>
<td>[14/May/2015:15:19:14]</td>
<td>+0000</td>
<td>POST /database-offline.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:19:14]</td>
<td>+0000</td>
<td>GET /index.php HTTP/1.1</td>
</tr>
<tr>
<td>127.0.1.1:08</td>
<td>192.168.72.137</td>
<td>-</td>
<td>-</td>
<td>[14/May/2015:15:19:17]</td>
<td>+0000</td>
<td>GET /index.php HTTP/1.1</td>
</tr>
</tbody>
</table>
Clean up

- Remove the columns

```r
apachelogs$Zone <- NULL
```
Packages

• Lots of functionality not delivered in the basic R install
• Bring on the packages
• Where can I find packages?
  • R Cran or Bioformatics or Github

  • install.packages("ggplot2")
  • library(ggplot2)
Is that all?

But wait... there's more!
Visualize

• Ggplot2 allows for plotting information in a graph
Let’s try it!

• Back to our web logs
• What would be interesting to graph?
  • How about Remote Hosts and Bytes? Why Not?

```r
p <- qplot(Remote.Host, Bytes, color = Bytes, data = apachelogs)
```

But we need to clean it up a bit as always:
```
p + theme(axis.text.y=element_text(hjust=0, angle=0), axis.text.x = element_text(hjust=0, angle=90))
```
And we now have value
Shiny

• Let’s get creative!
• Shiny allows us to build our own dashboard
• R programs embedded into a web page
• Prediction algorithms – Shiny can call your algorithm and display the results
• Uses bootstrap (looks nice and mobile friendly)
We can build Web Apps!

![Basic widgets](image-url)

- **Buttons**: Action
- **Single checkbox**: Choice A
- **Checkbox group**: Choice 1, Choice 2, Choice 3
- **Date range**: 2014-01-24 to 2014-01-24
- **File input**: Choose File
- **Help text**: Note: help text isn’t a true widget, but it provides an easy way to add text to accompany other widgets.
- **Radio buttons**: Choice 1, Choice 2, Choice 3
- **Select box**: Choice 1
- **Sliders**: Value 70
- **Numeric input**: Value 1
- **Text input**: Enter text...

![Histogram of x](image-url)
Case Study with R

• Again understand what your log format is
  • Know how you want to organize your data
  • Know what field headers they contain
  • Cleaning up your data can be tedious but worth it
  • There is much more to cleaning up the data than time allows
• R allows for RegEx’s,
Now Let's Maximize!

- Merge multiple data sets into one
- Clean out the garbage data
Tidy it up!

• How about this scenario?
  • Web application is suspected of being compromised?
    • What do we need to investigate?
      • Web Application Logs
      • Web Server Logs
      • Firewall Logs
      • Server Logs
      • What other logs are available?
Tidyr & dplyr

• The tidyr package makes it easy to reshape the layout of your data sets while retaining the relationships embedded in the data

• Makes your data “Tidy”

• Group your data with dplyr
Putting it all together

R allows us to pull the data directly from the sources
  - pull out the interesting information
  - create a script for the following:
    - reading the logfiles
    - pulling data (website, web crawling of data)
  - once data sets are pulled we need to clean them (remove columns, null data, unnecessary fields)
  - next script them to merge into one giant data set

- Factors to consider
  - many to many relationships
  - need to understand data to validate the merges and joins
  - multiple sets of code for graphics and visualization
End Results

- One massive data set that can be scripted, searched, and visualized
- Create algorithms to determine normal
- Show us the outliers, strange data, things not expected
- Activity of certain data sets
Data Exploration
How to Apply in Real World?

What if we were to take data from a bad reputation IP list and map it?
Yes We Can!

By using libraries such as ggplot, lattice, googleVis, ggmap and calling the URL we can download a reputation list and plot the locations on the map!
Baselines

• Baseline 7 days Database logs

• Take 1 hr of SQL Queries or 1 Day or 1 Week

  - TimeStamps
  - Server Type (which servers accessed the most)
  - Client IP / Server IP
  - DB Usernames
  - Source Program (to help identify client source)
  - SQL query
How about Netflow data

Top Talkers
- Who is talking to whom?
- what date/time
- volume

Bottom Talkers

Can we build our own SIEM?
- Live Data vs Archived Data issues
Feed your animal

Behavior based analysis
Recon analysis
Indicators of Compromise
Vulnerability Scanning
Unlimited Possibilities
How does this scale?

• It won’t always scale on your desktop
• Good for incident response analysis
• Long term need to move to big data Hadoop type solution
• Big R runs on Hadoop
cheatsheets

- Plenty of cheatsheets available from Rstudio
  - http://www.rstudio.com/resources/cheatsheets/
- R Dir
- R Bloggers
Thank you!

Questions?