R: A Proposed Analysis and Visualization Environment for Network Security Data

Joshua McNutt
CERT Network Situational Awareness Group,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
jmcnutt@cert.org

Abstract

The R statistical language provides an analysis environment which is flexible, extensible and analytically powerful. This paper details its potential as an analysis and visualization interface to SiLK flow analysis tools as part of a network situational awareness capability.

1 Introduction

The efficacy of network security analysis is highly dependent upon the data interface and analysis environment made available to the analyst. The command line seldom offers adequate visual displays of data, while many GUI designs necessarily limit the query specificity afforded at the command line. This paper proposes the use of R, a statistical analysis and visualization environment, for interfacing with flow data. R is a complete programming language and, consequently, is highly extensible. Its built-in analysis and visualization capabilities provide the analyst with a powerful means for investigating and modeling network behavior.

2 R! What is it good for?

R is a language and environment for statistical computing and graphics used by statisticians worldwide. It is syntactically very similar to the S language which was developed at Bell Laboratories (now Lucent Technologies). Unlike S, R is available as free software under the terms of the Free Software Foundation's GNU General Public License in source code form. Additional details are provided by the R Project for Statistical Computing (http://www.r-project.org/). The website also provides links to documentation and program files for downloading. Supported platforms include Windows, Linux and MacOS X.

R is an object-based environment which can run interactively or in batch mode. It has the ability to generate publication-quality graphical displays on-screen or for hardcopy. Users can write scripts and functions which leverage the programming language's many features, including loops, conditionals, user-defined recursive functions and input/output facilities. For computationally-intensive tasks, C and Fortran code can be linked.

There are a handful of packages supplied with the R distribution covering virtually all standard statistical analyses. Many more packages are available through the Comprehensive R Archive Network (CRAN), a family of Internet sites covering a very wide range of modern statistical methods.

3 SiLK Tools

The suite of command line tools known as the System for Internet-Level Knowledge (SiLK) are used for the collection and examination of Cisco NetFlow version 5 data. The CERT Network Situational Awareness (NetSA) Team wrote SiLK [1] for the purpose of analyzing flow data collected on large volume networks. Flow data provides summaries of host communications providing a comprehensive view of network traffic.

The SiLK analysis tools provide Unix-like commands with functionality that includes selecting (a.k.a. filtering), displaying (ASCII output), sorting and summarizing packed binary flow data. Multiple commands can also be piped together for complex filtering. In this paper, we utilize the tools *rwfilter* (to select the data) and *rwcount* to generate binned time series of flow records, bytes and packets and feed the results into R for analysis. Further details on the functionality of SiLK can be found in [1].

4 Motivation: Command Line versus GUI

Many experienced users enjoy the query specificity afforded by the command line. But, in order to visualize

R Objects	
Object	**Description**
vector	ordered collection of numbers
scalar	single-element vector
array	multi-dimensional vector
matrix	two-dimensional array
factor	vector of categorical [2] data
data frame	matrix-like structures in which the columns can be of different types (e.g., numerical and categorical variables)
list	general form of vector in which the various elements need not be of the same type, and are often themselves vectors or lists. Lists provide a convenient way to return the results of a statistical computation.
function	an object in R which manipulates other objects

Table 1: Data object types in R

their data, they must make do with a third-party graphing program. They often do not favor a graphical user interface because their options for both queries and visualization tend to become more limited. What we hope to provide with the R interface is a preservation of command line control with the added features of integrated visualization and analysis. Essentially, we would describe it as an enhanced command line experience, but it also provides the analyst with all of the benefits of the R language's object-based workspace model.

5 R Data Manipulation

5.1 R Data Objects

Every entity in the R environment is an object. Numeric vectors, ordered collections of numbers, are the simplest and most common type of object, but there are many others. See Table 1 for a description of the object types.

In this paper, our example uses a data frame to store our data. The data frame object is a very flexible matrix-like entity which, unlike a matrix, allows the columns to be of different types.

5.2 SiLK Data Access

It should be noted that while we use R to interface with SiLK, virtually any command-line tool could be used with R. Also, R has multiple SQL database interface libraries. Many methods exist for interfacing with data stores. We detail below the R-SiLK interface being used at this time.

Within R, wrapper functions tied to specific tools in the SiLK suite read in the user-specified SiLK command line as a text-string parameter. The wrapper function makes a system call to the computer running the flow tools. Then, using a standard R data input function, the wrapper function reads in the ASCII output of the command line call. The results of the wrapper function call are assigned to a list object in R. Each element of that list represents a different analysis result, e.g. a matrix of the data, summary statistics, etc. Subsequent analysis and visualization operations can then be applied to that output object or any of its elements.

5.3 R Workspace

All objects are located in the user's workspace which can be saved at the conclusion of the R session and restored at the start of the next session. The command *history()* produces a list of all commands submitted to R by the user.

5.4 Analysis Capability

From simple summary statistics to advanced simulations, the R platform provides functions, extension packages (available through CRAN) and visualization capabilities appropriate to a wide range of flow analysis tasks. The object-based nature of the R environment makes it a useful platform for the network security analyst. Objects from different analyses can be preserved in the user's workspace for comparison purposes. Also, rapid prototyping of new analysis tools is possible due to the wealth of built-in capabilities and the ease with which new functions can be written.

The CERT/NetSA Team has used R for a variety of analysis tasks, from logistic regression to robust correlation analysis. We have used its SQL interface functionality to access hourly roll-ups of flow data summarized by port and protocol from a special database created specifically for port analysis. This has made it possible to study temporal correlations in port activity and identify ports which are exhibiting substantial volumetric changes.

5.5 Graphing Capability

One of the most important features of R is its ability to create publication quality graphical displays. R has a huge set of standard statistical graphs, stemplots, boxplots, scatterplots, etc. Extension packages are available for more advanced 3D plotting and highly-specialized display types. The advantage for the analyst running R in interactive mode is the ability to make slight changes

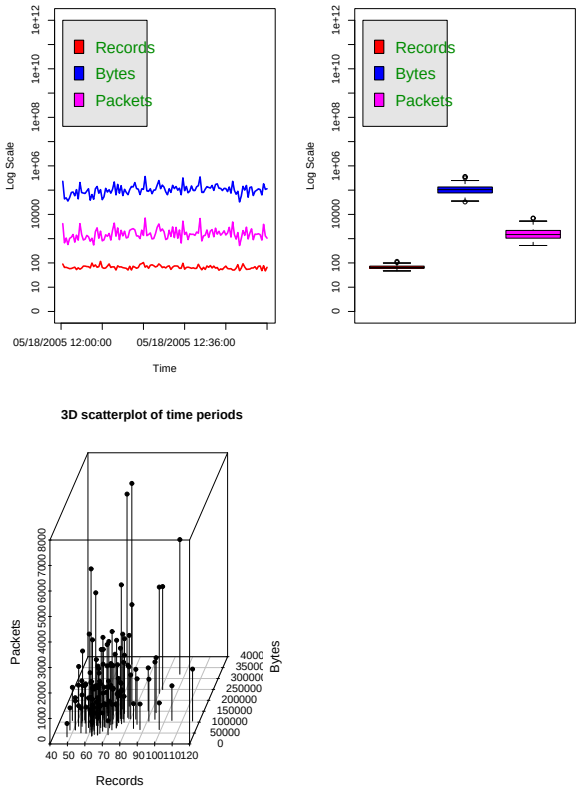

Figure 1: Graphical output of rwcount.analyze()

to the SiLK query and quickly visualize those changes in a newly drawn graph. Given the flexibility of its graphical facilities, R is also an ideal environment for advanced analysts to perform visualization prototyping.

6 R-SiLK wrapper function prototype: *rwcount.analyze()*

Our first proof-of-concept SiLK interface function is the wrapper *rwcount.analyze()* which calls the SiLK tool *rwcount*. Details of this wrapper function are provided in Table 2. The function has two input parameters, *command* and *plot*. The parameter command is a text string which is assigned a SiLK command line call to *rwcount*, which returns binned time series of records, bytes, and flows. The other input parameter, *plot*, determines whether a graphical display will be generated at runtime. The default is *plot*=TRUE. The visualization provided in our prototype includes three plots: a time series plot, side-by-side boxplots, and a 3D scatterplot of the data. Figure 1 provides an example of the graphical output generated by *rwcount.analyze()*.

When *rwcount.analyze()* is called, its output is assigned to a list object in R. The list it generates contains five elements: *data*, *command*, *stats*, *cor*, and *type*. These elements are defined in Table 2.

A sample R session using *rwcount.analyze()* to examine FTP traffic is provided below. The parameter command is assigned a SiLK command line. In our example, we specify TCP traffic (−−proto=6) directed at destination port 21 (−−dport=21) for the hour between noon and 1 p.m. on May 18, 2005. Those specifications are provided to *rwfilter* via switches, and the selected flows (in binary, packed format) are piped into *rwcount* where we have specified a bin size of thirty seconds (−−bin-size=30). The output of *rwcount* consists of the time series of bytes, records and packets which are read into a data frame object in R. This data frame is also an element in the output list object returned by *rwcount.analyze()*.

In this example, the output list returned by the function is assigned to *obj*. The list of object elements are printed with the function *names()* and correspond to the items in Table 2. As an example of automated analysis that can be returned in a results object, the correlation matrix of the series is found in *obj$cor*. This output shows that bytes, records and packets are highly correlated with each other ($\rho > .99$). Since *obj$data* is a data frame of the three time series, we can print the records field by typing *obj$data$Records*. This is one of the time series plotted in Figure 1.

```
> obj <- rwcount.analyze(command=
"rwrun rwfilter
--start-date=2005/05/18:12:00:00
--proto=6
--dport=21
--print-file
--pass=stdout |
rwcount
--bin-size=30",
plot=TRUE)

> names(obj)
[1] "data" "command" "stats" "cor"
[5] "type"

> obj$cor
            Records  Bytes    Packets
Records    1.0000   0.9944   0.9951
Bytes      0.9944   1.0000   0.9964
Packets    0.9951   0.9964   1.0000

> obj$data$Records
                     Records
05/18/2005 12:00:00   76218
05/18/2005 12:05:00   73374
```

rwcount.analyze() details	
Input Parameters	
Parameter	**Description**
command	SiLK command line text string
plot	Logic element determines whether R will perform runtime plotting
Output List Elements	
List Element	**Description**
data	Data frame containing rwcount time series for Bytes, Records and Packets
command	Same as input parameter description
stats	Summary statistics for Bytes, Records and Packets
cor	Correlation matrix for Bytes, Records and Packets
type	Text string to indicate which wrapper function generated this object

Table 2: rwcount.analyze() function description

```
05/18/2005 12:10:00   55743
...
```

7 Analyst Benefits

One of the advantages of R is its potential for rapid analysis prototyping. A user can very quickly write functions that generate a slew of experimental analysis results describing a host, a subnet, or traffic volumes. Each result can be included in the function's output list and evaluated. Analysis results which prove useful can be quickly integrated and become standard output elements.

In analytical work, the ability to label preliminary results objects provides the investigator with a facility for generating an audit trail. In R, this labeling is performed by the addition of object elements which describe the object to either the analyst or other functions which will operate on the object. By default, *rwcount.analyze()* returns the elements *type* and *command*. The element *type* can be used to describe the object to other functions. For example, a generic graphing function (perhaps called *rw.visualize()*) would read in an object and determine how it should be displayed based upon its *type*. The element *command* describes to the user how the object was created by storing the SiLK command. Additional elements can also be added to existing objects. For instance, a user may wish to attach a comment (e.g. "Surge in host count lasted for 6 hours") to an object by adding a text string element.

Since objects are preserved when the users save their workspace in R, comparison with objects from future analyses is very simple. Also, the user can graph objects from a previous analysis side-by-side with new results.

We believe the experienced analyst will leverage the enhanced command line experience, fast visualization and rapid analysis prototyping. For analyses requiring longer data pulls, R can also serve as an integrated scripting and analysis environment.

We envision a hierarchy of analysis functions. At the lowest level would be functions like *rwcount.analyze()* which use a SiLK command line call as a parameter. A function at the next level of the hierarchy would allow a user to specify criteria of interest via function parameters (e.g. dport=80, proto=6). This function would both generate the necessary SiLK command line and submit it to *rwcount.analyze()* for processing. Using these functions, novice analysts unacquainted with the SiLK command line would be able to perform real analysis tasks immediately. These functions could also be used for learning purposes since the SiLK command line needed for the query is provided in the output object.

8 Future Work

Our wrapper function *rwcount.analyze()* is merely a proof-of-concept prototype of an interface between R and SiLK. Next steps include the development of additional wrapper functions, making further improvements to *rwcount.analyze()*, and developing a generic visualization scheme that reads the *type* field in an output object to determine the appropriate display.

9 Conclusion

This paper has introduced the reader to R, demonstrating an overlap between its capabilities and the needs of network security analysts. R provides a truly integrated environment for data analysis and visualization. Further, the ability to interface with SiLK flow analysis tools and other data storage formats makes it an ideal environment for enhancing and extending a network situational awareness capability.

References

[1] CARRIE GATES, MICHAEL COLLINS, E. A. More netfbw tools: For performance and security. In *LISA XVIII* (2004), pp. 121–131.

Notes

[1] http://silktools.sourceforge.net/

[2] We are using "categorical" here to describe string character data (e.g. "male" versus "female").