Big Data Security Analysis Approach Using Computational Intelligence Techniques in R for Desktop Users

Nitin Naik[1], Paul Jenkins[1], Nick Savage[2] and Vasilios Katos[3]

[1]Defence School of Communications and Information Systems, Ministry of Defence, United Kingdom
[2]School of Computing, University of Portsmouth, United Kingdom
[3]Department of Computing and Informatics, Bournemouth University, United Kingdom
Email: {nitin.naik100, paul.jenkins683}@mod.uk, nick.savage@port.ac.uk, vkatos@bournemouth.ac.uk

***Abstract*—Big Data security analysis is commonly used for the analysis of large volume security data from an organisational perspective, requiring powerful IT infrastructure and expensive data analysis tools. Therefore, it can be considered to be inaccessible to the vast majority of desktop users and is difficult to apply to their rapidly growing data sets for security analysis. A number of commercial companies offer a desktop-oriented big data security analysis solution; however, most of them are prohibitive to ordinary desktop users concerning their cost and the requirement for powerful IT infrastructure. This paper presents an intuitive and inexpensive big data security analysis approach using Computational Intelligence (CI) techniques for Windows desktop users, where the combination of Windows batch programming, EmEditor and R are used for the security analysis. The simulation is performed on a real dataset with more than 10 million observations, which are collected from Windows Firewall logs to demonstrate how a desktop user can gain insight into their abundant and untouched data and extract useful information to prevent their system from current and future security threats. This CI-based big data security analysis approach can also be extended to other types of security logs such as event logs, application logs and web logs.**

***Keywords*—Big Data, Security Analysis, Computational Intelligence Techniques, CI, R, Desktop User, Windows Firewall Logs**

I. Introduction

Security analysis is becoming an increasingly complex task for desktop users due to the enormous data generated from different security tools in the form of firewall logs, event logs, application logs, web logs and many other security logs. The efficient handling and processing of this collected data require large system resources and powerful analysis tools. However, traditional systems and tools are not capable of handling and analysing these large unstructured datasets. In the absence of the proper processing mechanism for these large datasets, these valuable datasets may become useless and a resource overhead for the other important applications. Therefore, desktop users require an easy to implement and inexpensive big data security analysis approach that meets their data processing requirements within their limitations. However, most of the security analysis solutions are not affordable to the ordinary desktop user due to their costs and the requirement for powerful systems. Another issue is the sophisticated use of these complex tools, whereas, many ordinary users are untrained IT users or reluctant to receive long and complex IT applications training. Therefore, desktop users require a relatively simple, economical and resource efficient data analysis approach. **R** is an open-source data analysis tool consisting of various CI packages for advanced data analysis. However, it requires a basic understanding of statistics which is desirable for any data analysis. **R** may not be suitable for the data collection or cleaning functions but it could be used for various analyses with some additional supporting tools.

This paper presents an intuitive and inexpensive security analysis approach using CI techniques in **R** for Windows desktop users. The choice of Windows desktop was as result of its popularity, where the Microsoft Windows operating system, is installed on has approximately 70% of the computer operating system market [1]. Therefore, there are large populations of Windows desktop users. If Windows desktop users can find or design an easy to implement and inexpensive big data security analysis solution supported by their system, then they can analyse very large security logs to extract meaningful security information to improve their systems security, making it more robust [2], [3]. In this proposed security analysis approach, the combination of Windows batch programming, EmEditor (which can be replaced with any powerful editor) and **R** are used for analysis purposes. **R** hosts several CI packages related to artificial neural networks, evolutionary algorithms, fuzzy systems and hybrid intelligent systems for designing intelligent systems. This security analysis approach involves several stages, where data collection and merging are performed by using a Windows batch script; data cleaning and editing are carried out by using EmEditor; and finally, **R** is used for structuring the data, performing analysis using CI techniques, visualising and interpreting the results. The experimental simulation is based on a real dataset of $1,006,889,160$ bytes (1.01 GB) with more than 10 million observations, which are collected from the Windows Firewall logs during the log recording process for a 30 day period. Subsequently, security analysis is carried out on the collected Windows Firewall logs to demonstrate how a desktop user can gain insight into their abundant and untouched data and extract useful information to prevent their system from current and future security threats [2]. This CI-based big data security analysis approach can also be extended to other types of security logs such as event logs, application logs and web logs.

The remainder of this paper is organised as follows: Section II explains the theoretical background of data analysis tool **R**, Windows Firewall and fuzzy reasoning; Section III illustrates the design and implementation process of the proposed security analysis approach including its various stages: collecting and merging logs, cleaning and editing logs, converting text logs into an **R** table structure, analysing **R** datasets using CI techniques, and visualising and interpreting the results; Section IV explains the big data scalability of this approach for desktop users. Finally, Section V concludes the paper and suggests some future areas of extension.

II. Theoretical Background

This section presents the background information about the data analysis tool **R**, Windows Firewall and fuzzy reasoning.

A. ***R***

R is an open-source statistical computation and data visualisation software tool. It is the result of collaboration of a large team of developers, researchers, statisticians and data scientists from around the world. **R** is available for all the main operating systems such as UNIX, Windows and MacOS platforms. **R** comprises data handling facilities, a superior mechanism for matrix computations, a plethora of data analysis and graphical packages, and a simple programming language [4]. The most powerful feature of **R** is subsumption i.e. its support to external packages. Currently, **R** has incorporated around 5000 packages through the CRAN family of Internet sites [5]. **R** also hosts several CI packages related to artificial neural networks, evolutionary algorithms, fuzzy systems and hybrid intelligent systems for designing intelligent systems. Therefore, combining **R** with some data collecting and cleaning tools could facilitate a potential data analysis solution for desktop users. **R** is used as a computational tool for routine statistics production by many official statistics agencies. Besides official statistics, it is used in many other sectors such as finance, retail, manufacturing, science, and academic research, which is making it a popular tool among statisticians and researchers [4].

B. Windows Firewall

Microsoft embedded the firewall utility in Windows operating systems since Windows XP SP2 and is now available with all versions of Windows. Windows Firewall with "advanced security" features is a stateful firewall that examines and filters all packets for IPv4 and IPv6 traffic. The packet filtering process is based on the user or administrator-defined rules and on that basis it allows or blocks the network traffic. The firewall automatically blocks all incoming traffic unless it is a response to a request by the host or it is specifically allowed by writing a firewall rule. Windows Firewall can also be configured with "advanced security" for a specific port number, application name, service name, or other criteria based traffic; then this traffic can be allowed explicitly [6]. These features are designed for advanced users who need to manage network security in an enterprise environment. It is not often intended for the use in home networks.

C. Fuzzy Reasoning

Fuzzy reasoning is the process of deriving logical conclusions from an existing fuzzy rule base [7]. It mimics the ability of the human mind to summarize data and focus on decision-relevant information [8]. Fuzzy reasoning is more effective and useful for those systems where a system cannot be defined in precise mathematical terms or models due to uncertainties, unpredicted dynamics and other unknown phenomena [9]. In network security, much of the information and traffic data is incomplete and imprecise in nature. Therefore, fuzzy reasoning is comparatively more suitable than other types of reasoning approaches [10], [11], [12], [13]. Fuzzy reasoning is based on a fuzzy rule base, and it can be derived by subject matter experts or extracted from data through a rule induction process. If the fuzzy rule base is a dense rule base then, any rule inference method such as Mamdani inference [14] or Takagi-Sugeno inference [15] can be used.

III. Security Analysis of Windows Firewall Logs Using Computational Intelligence Techniques in R

The security analysis for desktop users is a challenging task due to the limitation of system resources and technical IT skills. Therefore, this section presents the design and implementation of the proposed CI-based big data security analysis approach for desktop users for performing security analysis within their limitations. This security analysis only focuses on Windows desktop users. The experiment is carried out on the Windows 7 operating system and desktop with configuration (Processor=Intel Core i7 3.0 GHz (4 cores), RAM=16 GB, L2 Cache=8 MB, Ethernet=100 Mbps). This security analysis requires two software tools: any powerful text editor (such as EmEditor in this implementation) and **R** (with RStudio IDE). Unlike the other security analysis, where prior technical training is necessary, here any user with basic knowledge of statistics and elementary IT skills can conduct the security analysis without any prior technical training. This desktop-oriented security analysis approach has several stages as shown in Fig. 1. The description of the various stages are as follows:

A. Collecting and Merging Windows Firewall Logs using Windows Batch Script

The Windows Firewall logs are recorded in the "pfirewall.log" by default. The maximum size of the "pfirewall.log" file is 4096 KB. After exceeding this limit, it saves the logs in a backup file called "pfirewall.log.old" of 4096 KB. Both log files do not grow beyond this size, and when the "pfirewall.log" file exceeds the maximum limit again, the old log entries are deleted to make room for the newly created ones. In this security analysis of Windows Firewall logs, a reasonable log file was required to perform extensive analysis and that was created in the most simplest way by writing a Windows batch script as shown in Fig. 2. This batch script worked in the background during the period of the complete experiment. Initially, the "mergedLog" file was created with one line of a header containing all the 17 default variables of the "pfirewall.log" file. Finally, the "mergedLog" file of $1,006,889,160$ bytes (1.01 GB) with more than 10 million observations was obtained during the gradual log recording process for 30 day period.

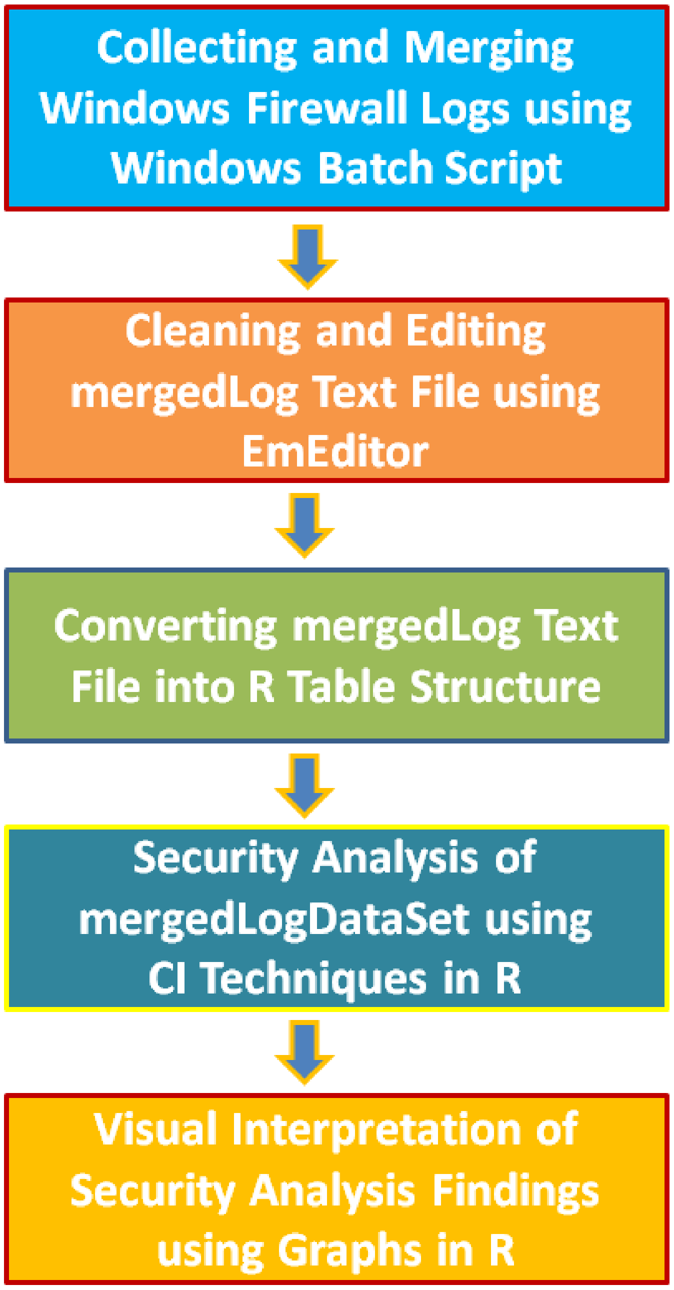
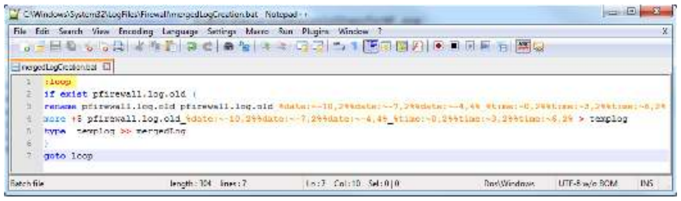

Collecting and Merging Windows Firewall Logs using Windows Batch Script
Cleaning and Editing mergedLog Text File using EmEditor
Converting mergedLog Text File into R Table Structure
Security Analysis of mergedLogDataSet using CI Techniques in R
Visual Interpretation of Security Analysis Findings using Graphs in R

Fig. 1. Stages of big data security analysis approach for analysing Windows Firewall logs

*B. Cleaning and Editing **mergedLog** Text File using EmEditor*

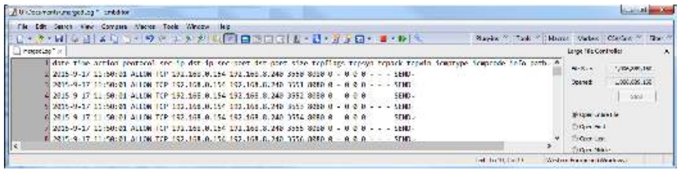
Normally, the collected data may be incorrect, incomplete, improperly formatted, or duplicated and require cleaning and editing for removing these impurities [16], [17]. While the log file was a simple text file with no major cleaning issues, only a powerful data editor (EmEditor) was used rather than a proper cleaning tool. Notepad++ and other desktop editors were not capable of handling a very large text file, whereas EmEditor could easily handle a file size up to 248 GB [18]. For this analysis, the "mergedLog" file (above 1 GB) was cleaned and edited using EmEditor as shown in Fig. 3. The "mergedLog" file was created in a way that the first five lines of the "pfirewall.log.old" were removed every time before


C:\Windows\System32\LogFiles\Firewall\mergedLogCreation.bat - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
mergedLogCreation.bat
Batch file
length : 104 lines : 7
Ln : 7 Col : 10 Sel : 0 | 0
Dos\Windows
UTF-8 w/o BOM
INS

```
:loop
if exist pfirewall.log.old (
rename pfirewall.log.old pfirewall.log.old_%date:~-10,2%%date:~-7,2%%date:~-4,4%_%time:~0,2%%time:~3,2%%time:~6,2%
more +5 pfirewall.log.old_%date:~-10,2%%date:~-7,2%%date:~-4,4%_%time:~0,2%%time:~3,2%%time:~6,2% > templog
type  templog >> mergedLog
)
goto loop
```

Fig. 2. Windows batch script for creating a mergedLog file of Windows Firewall logs

copying it to the "mergedLog" file. However, the "mergedLog" file still needed to be checked manually for various purposes and aligned as per the requirement of **R**, because **R** can only accept the file in a table format with the correct alignments. If the "mergedLog" file did not fit in the **R** format, then **R** could generate an error message, and data could not be imported. Therefore, the major cleaning and editing task in "mergedLog" file was to check spaces between the two fields and align all lines including the header and last line if required.


D:\Documents\mergedLog* - EmEditor
File Edit Search View Compare Macros Tools Window Help
mergedLog*
Large File Controller
Opened

```
date time action protocol src-ip dst-ip src-port dst-port size tcpflags tcpsyn tcpack tcpwin icmptype icmpcode info path
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3550 8080 0 - 0 0 0 - - - SEND-
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3551 8080 0 - 0 0 0 - - - SEND-
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3552 8080 0 - 0 0 0 - - - SEND-
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3553 8080 0 - 0 0 0 - - - SEND-
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3554 8080 0 - 0 0 0 - - - SEND-
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3555 8080 0 - 0 0 0 - - - SEND-
2015-9-17 11:50:01 ALLOW TCP 192.168.0.154 192.168.8.240 3556 8080 0 - 0 0 0 - - - SEND-
```

Fig. 3. Cleaning and editing mergedLog text file in EmEditor

*C. Converting **mergedLog** Text File into R Table Structure*

R supports a table-kind of data structure based on the **R** *data frame*. Therefore, the data frames are the fundamental data structure in **R**. The read.table() reads a file in table format and creates a data frame from it. The syntax of this function is shown in Equation 1.

$$\begin{aligned} &data\_frame\_name = \\ &\quad read.table(file\_name, header = F/T, sep = \text{“ ”}) \end{aligned} \tag{1}$$

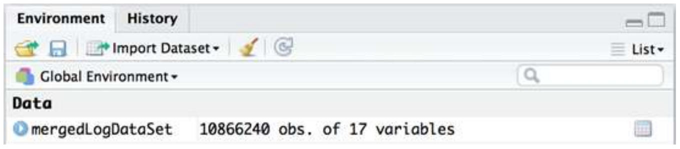
where the "header" is a logical value indicating whether the file contains variable names as its first line and "sep" is the field separator character. In this implementation, the merged log file "mergedLog" was converted into a table-kind of data structure using the read.table() function as shown in Fig. 4.

```
> mergedLogDataSet =
    read.table("C:/Documents/mergedLog", header=T, sep="")
```

Fig. 4. Creation of mergedLogDataSet in R

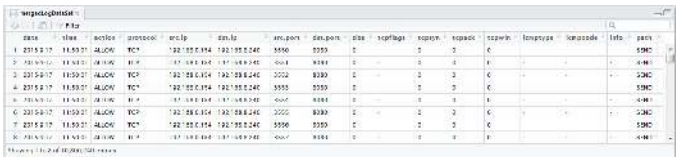
If the structure, header and content of the merged log file were accurate, then it created a dataset called "mergedLog-DataSet" in **R** as shown in Fig. 5. This dataset was displayed in the "Environment and History Pane" of *RStudio IDE* and contained 10866240 observations and 17 variables similar to the firewall header variables. The actual table structure of the "mergedLogDataSet" (see Fig. 6) could be seen in the "Script Editor Pane", usually at the opposite side of the "Environment and History Pane" in *RStudio IDE*.

The structure-related information of this mergedLog-DataSet could be seen by "str" command as shown in Fig.



```
mergedLogDataSet    10866240 obs. of 17 variables
```

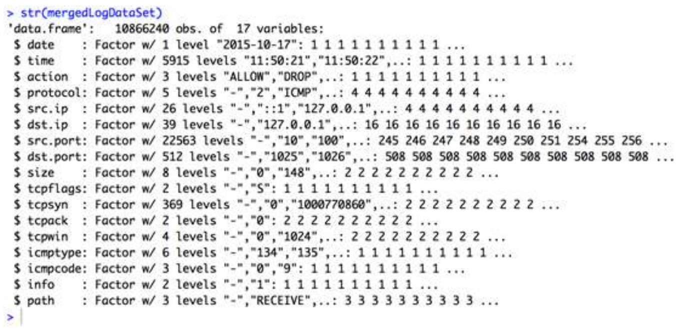
Fig. 5. Created mergedLogDataSet with number of observations and variables



	date	time	action	protocol	src.ip	dst.ip	src.port	dst.port	size	tcpflags	tcpsyn	tcpack	tcpwin	icmptype	icmpcode	info	path
1	2015-10-17	11:50:21	ALLOW	TCP	192.168.0.154	192.168.0.240	5550	8080	0		0	0	0				SEND
2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3	2015-10-17	11:50:21	ALLOW	TCP	192.168.0.154	192.168.0.240	5552	8080	0	-	0	0	0	-	-	-	SEND
4	2015-10-17	11:50:21	ALLOW	TCP	192.168.0.154	192.168.0.240	5553	8080	0		0	0	0				SEND
5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	2015-10-17	11:50:21	ALLOW	TCP	192.168.0.154	192.168.0.240	5555	8080	0	-	0	0	0	-	-	-	SEND
7	2015-10-17	11:50:21	ALLOW	TCP	192.168.0.154	192.168.0.240	5556	8080	0		0	0	0				SEND
8	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Fig. 6. Table structure of mergedLogDataSet in **R**

7. This "mergedLogDataSet" was directly created from the Windows Firewall merged log text file; therefore, the data types of all the 17 variables were "factor". This data type information was really crucial for most of the statistical analysis because the "factor" was a categorical data in **R** and for many computing models, it needed to be converted into numerical data.



```
> str(mergedLogDataSet)
'data.frame':   10866240 obs. of  17 variables:
 $ date     : Factor w/ 1 level "2015-10-17": 1 1 1 1 1 1 1 1 1 1 ...
 $ time     : Factor w/ 5915 levels "11:50:21","11:50:22",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ action   : Factor w/ 3 levels "ALLOW","DROP",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ protocol: Factor w/ 5 levels "-","2","ICMP",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ src.ip   : Factor w/ 26 levels "-","::1","127.0.0.1",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ dst.ip   : Factor w/ 39 levels "-","127.0.0.1",..: 16 16 16 16 16 16 16 16 16 16 ...
 $ src.port: Factor w/ 22563 levels "-","10","100",..: 245 246 247 248 249 250 251 254 255 256 ...
 $ dst.port: Factor w/ 512 levels "-","1025","1026",..: 508 508 508 508 508 508 508 508 508 508 ...
 $ size     : Factor w/ 8 levels "-","0","148",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ tcpflags: Factor w/ 2 levels "-","S": 1 1 1 1 1 1 1 1 1 1 ...
 $ tcpsyn   : Factor w/ 369 levels "-","0","1000770860",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ tcpack   : Factor w/ 2 levels "-","0": 2 2 2 2 2 2 2 2 2 2 ...
 $ tcpwin   : Factor w/ 4 levels "-","0","1024",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ icmptype: Factor w/ 6 levels "-","134","135",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ icmpcode: Factor w/ 3 levels "-","0","9": 1 1 1 1 1 1 1 1 1 1 ...
 $ info     : Factor w/ 2 levels "-","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ path     : Factor w/ 3 levels "-","RECEIVE",..: 3 3 3 3 3 3 3 3 3 3 ...
>
```

Fig. 7. Created mergedLogDataSet with number of observations and variables

D. Security Analysis of *mergedLogDataSet* using CI Techniques in *R*

The main aim of the security analysis could be different for different users depending on their requirements. This particular analysis focuses on summarising the "mergedLogDataSet" for extracting vital information, deciding the security status of the desktop using the null hypothesis and binomial analysis, investigating the abnormalities in details based on targeted protocols and IP addresses, designing an intelligent system to predict the risk of attack and finally, graphical illustration of the security analysis findings. The sequence and details of the various analyses are as follows:

1) Windows Firewall Rules for Security Analysis: In this simulation, a few firewall rules were created for security analysis purpose. Subsequently, these rules-based traffic data were collected in the "pfirewall.log" file. Two inbound rules were created to block two particular computers for specific traffic as shown in Fig. 8. The first rule was created to block the computer with IP address 192.168.0.50 for only ICMP packets, and the second rule for the computer with IP address 192.168.0.51 for only TCP packets. Therefore, enough dropped activities could be recorded during the log generation period. Similarly, for stopping some activities at the host end (192.168.0.154), an outbound rule was also created to block the outgoing ICMP packets to other computers as shown in Fig. 9. These blocking rules generated enough "drop" traffic in the firewall log for security analysis.

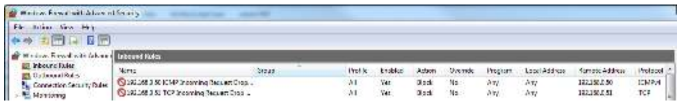


Fig. 8. Incoming ICMP and TCP Drop Rules for IP addresses 192.168.0.50 and 192.168.0.51 respectively

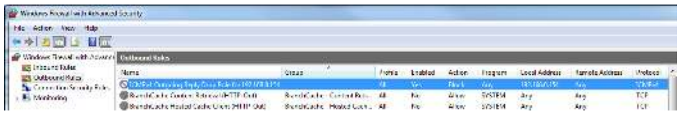
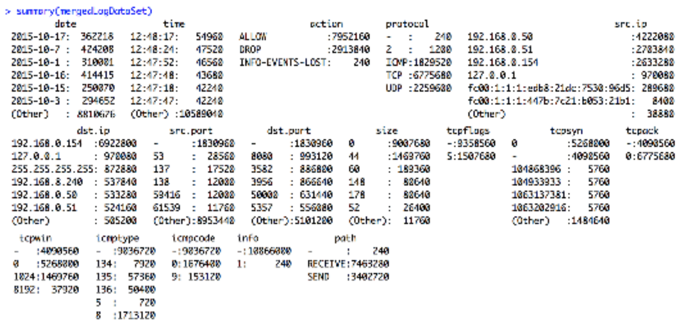


Fig. 9. Outgoing ICMP Drop Rule for the host IP address 192.168.0.154

2) Preliminary Statistical Analysis of Windows Firewall Log: The simplest data analysis command in **R** is the "summary" command that gives reasonable statistics about the given dataset. However, it may be insufficient for detailed investigations or to predict future trends. Thus, some advanced analysis packages may be needed depending on the nature of the study. In Fig. 10, the summary command shows few statistics about "mergedLog" file, which could be very useful for further investigations. This summary includes information related to the date, time, action, protocol, source address, destination address, source port, and destination port, which are quite clear and understandable.

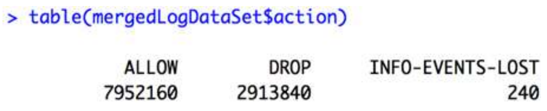


```
> summary(mergedLogDataSet)
        date              time                          action           protocol                        src.ip
 2015-10-17:  362218   12:48:17:   54960   ALLOW             :7952160   -   :    240   192.168.0.50                   :4222080
 2015-10-7 :  424208   12:48:24:   47520   DROP              :2913840   2   :   1200   192.168.0.51                   :2703840
 2015-10-1 :  310001   12:47:52:   46560   INFO-EVENTS-LOST:     240   ICMP:1829520   192.168.0.154                  :2633280
 2015-10-16:  414415   12:47:48:   43680                                 TCP :6775680   127.0.0.1                      : 970080
 2015-10-15:  250070   12:47:18:   42240                                 UDP :2259600   fc00:1:1:1:edb8:21dc:7530:96d5: 289680
 2015-10-3 :  294652   12:47:47:   42240                                                fc00:1:1:1:447b:7c21:b053:21b1:   8400
 (Other)   : 8810676   (Other) :10589040                                                (Other)                        :  38880
          dst.ip            src.port          dst.port          size          tcpflags           tcpsyn          tcpack
 192.168.0.154  :6922800   -      :1830960   -      :1830960   0      :9007680   -:9358560   0          :5268000   -:4090560
 127.0.0.1      : 970080   53     :  28560   8080   : 993120   44     :1469760   S:1507680   -          :4090560   0:6775680
 255.255.255.255: 872880   137    :  17520   3582   : 886800   60     : 189360               104868396 :    5760
 192.168.0.240  : 537840   138    :  12000   3956   : 866640   148    :  88640               104933933 :    5760
 192.168.0.50   : 533280   59416  :  12000   50000  : 631440   178    :  88640               1063137381:    5760
 192.168.0.51   : 524160   61539  :  11760   5357   : 556080   52     :  26400               1063202916:    5760
 (Other)        : 505200   (Other):8953440   (Other):5101200   (Other):  11760               (Other)   :1484640
  tcpwin          icmptype       icmpcode     info           path
 -   :4090560   -   :9836720   -:9836720   -:10866000   -      :    240
 0   :5268000   134:    7920   0:1676400   1:     240   RECEIVE:7463280
 1024:1469760   135:   57360   9: 153120                SEND   :3402720
 8192:  37920   136:   50400
                5  :     720
                8  :1713120
```

Fig. 10. Summary analysis of mergedLogDataSet

3) Null Hypothesis and Binomial Exact Analysis: The summary analysis presented is only superficial data, and the firewall log became gigantic. Therefore, for ordinary users, it is very difficult to extract meaningful security information from this firewall log. The first step for a user would be to decide whether the collected traffic data is close to the normal/ideal traffic level or not. Thus, a different statistical analysis is required to assess the current security status of the desktop. The null hypothesis in **R** is the simplest analysis to compare the statistical significance of the data without complicating the analysis with further details. The central action of any firewall is to allow or drop packets based on their rules. For this, the "table()" function displays a table of the counts at each combination of the factor levels. In Fig. 11, table() function displays and simplifies the firewall actions against all

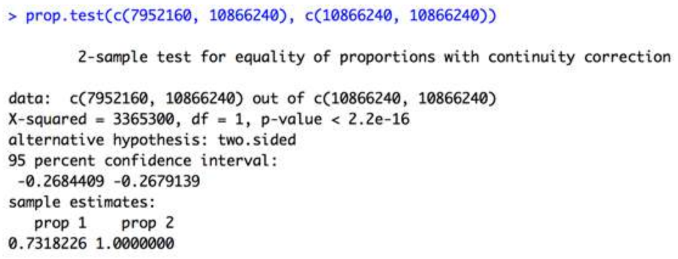
10866240 packets, where 7952160 packets are "allowed" and 2913840 packets are "dropped" out. Therefore, based on the number of packets allowed (successes), the null hypothesis is constructed to determine the collected traffic status/level.



```
> table(mergedLogDataSet$action)

    ALLOW        DROP    INFO-EVENTS-LOST
  7952160     2913840                 240
```

Fig. 11. Summary of allow, drop and lost packets

In the null hypothesis, the number of allowed packets (7952160) was compared with the ideal traffic condition when all the packets (10866240) could have allowed for checking the statistical similarity between the two samples so the desktop security level could be assessed with its ideal traffic condition. With the significance level = 0.05 and the level of confidence = 95%, the p-value given by the prop.test() function was $2.2e{-}16$ (i.e., p-value $<$ .Machine$double.eps in R) as shown in Fig. 12. This value $2.2 * 10^{-16}(0.00000000000000022)$ was effectively close to zero (actually numerically indistinguishable from 0) and much smaller than the value (0.05) of the significance level. Additionally, the value 0 did not lie within the confidence interval $(-0.2684409$ and $-0.2679139)$ as shown in Fig. 12. Therefore, the correlation was statistically significant, and the null hypothesis was rejected with the high degree of significance. This result stated that the desktop's current traffic condition was not normal and, thus, there was a need for further detailed investigation about the types of attacks/threats.

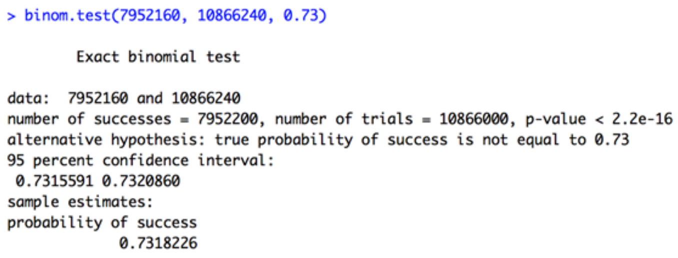


```
> prop.test(c(7952160, 10866240), c(10866240, 10866240))

	2-sample test for equality of proportions with continuity correction

data:  c(7952160, 10866240) out of c(10866240, 10866240)
X-squared = 3365300, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.2684409 -0.2679139
sample estimates:
   prop 1    prop 2
0.7318226 1.0000000
```

Fig. 12. Null hypothesis analysis to determine the desktop security status based on the allowed traffic

The results of the null hypothesis test were also verified precisely by the Binomial exact test in **R**. In Fig. 12, the probability of successes (in this case allowed packets) is $0.7318226 \approx 0.73$. Therefore, the Binomial exact value was calculated using the binom.test() function as shown in Fig.13. However, the p-value was the same as the previous p-value calculated by prop.test() function. Thus, both tests rejected the null hypothesis with the significance level = 0.05 and the level of confidence = 95%. Consequently, a further detailed analysis was required to obtain the nature of the risks and attacks to the desktop.

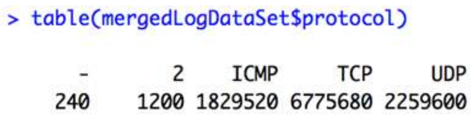
4) ICMP/TCP/UDP Packets and IP Address Analysis: The null hypothesis and binomial analysis led to the further investigation of the firewall log to identify the causes of security breaches. In the summary analysis results shown in Fig. 10, the protocols and IP addresses related information could be easily observed and useful to analyse the causes. Fig. 14 shows the summary table of the total ICMP, TCP, and UDP



```
> binom.test(7952160, 10866240, 0.73)

	Exact binomial test

data:  7952160 and 10866240
number of successes = 7952200, number of trials = 10866000, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.73
95 percent confidence interval:
 0.7315591 0.7320860
sample estimates:
probability of success
             0.7318226
```

Fig. 13. Binomial exact analysis to determine the desktop security status based on the allowed traffic

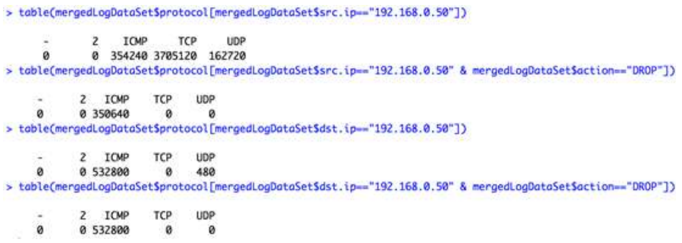
packets recorded in firewall logs over the simulation period. In Windows Firewall, the protocol options available are TCP, UDP, ICMP, and a protocol number for packets that are not TCP, UDP, or ICMP. Therefore, "2" is a protocol number and "-" is used for the lost packets (see Info-Events-Lost in Fig. 11).



```
> table(mergedLogDataSet$protocol)

      -       2    ICMP     TCP     UDP
    240    1200 1829520 6775680 2259600
```

Fig. 14. Summary of total ICMP, TCP, and UDP packets

Here, Figs. 15 and 16 show the detailed analysis of the two protocols ICMP and TCP, and two computer systems with IP addresses: 192.168.0.50 and 192.168.0.51. The system with IP address 192.168.0.50 was blocked for only ICMP packets during the experiment for the maximum period of time but not for the entire duration. Therefore, Fig. 15 shows almost all ICMP packets (350640) as the dropped packets where the source IP address was 192.168.0.50. Another rule was written for the host (192.168.0.154) to stop any outgoing ICMP packets; consequently, all 532800 packets were dropped, which were tried to send to the destination IP address 192.168.0.50. However, other TCP and UDP packets were allowed through the firewall. The system with IP address 192.168.0.51 was blocked

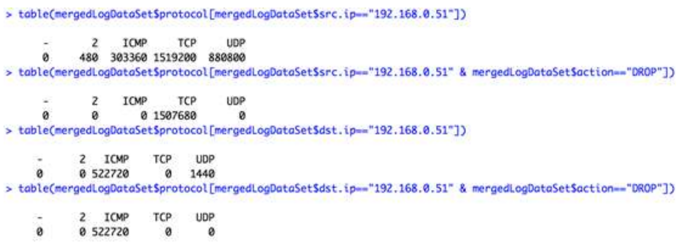


```
> table(mergedLogDataSet$protocol[mergedLogDataSet$src.ip=="192.168.0.50"])

      -       2    ICMP     TCP     UDP
      0       0  354240 3705120  162720
> table(mergedLogDataSet$protocol[mergedLogDataSet$src.ip=="192.168.0.50" & mergedLogDataSet$action=="DROP"])

     -      2   ICMP    TCP    UDP
     0      0 350640      0      0
> table(mergedLogDataSet$protocol[mergedLogDataSet$dst.ip=="192.168.0.50"])

     -      2   ICMP    TCP    UDP
     0      0 532800      0    480
> table(mergedLogDataSet$protocol[mergedLogDataSet$dst.ip=="192.168.0.50" & mergedLogDataSet$action=="DROP"])

     -      2   ICMP    TCP    UDP
     0      0 532800      0      0
```

Fig. 15. Summary of protocols for IP address 192.168.0.50

for only TCP packets during the experiment for the maximum period of time but not for the entire duration. Therefore, Fig. 16 shows almost all TCP packets (1507680) as the dropped packets where the source IP address was 192.168.0.51. The previously written rule for the host (192.168.0.154) to stop any outgoing ICMP packets also enforced here; consequently, all 522720 packets were dropped, which were tried to send to the destination IP address 192.168.0.51. However, other TCP and UDP packets were allowed through the firewall.

5) Designing the Fuzzy Inference System for Predicting Risk of Attack: An analysis often requires the modelling

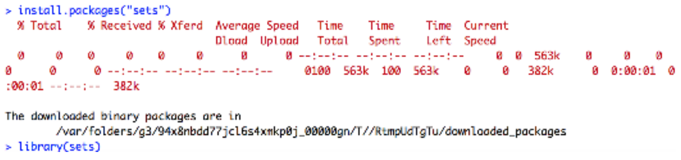


```
> table(mergedLogDataSet$protocol[mergedLogDataSet$src.ip=="192.168.0.51"])

      -       2    ICMP     TCP     UDP 
      0     480  303360 1519200  880800 
> table(mergedLogDataSet$protocol[mergedLogDataSet$src.ip=="192.168.0.51" & mergedLogDataSet$action=="DROP"])

      -       2    ICMP     TCP     UDP 
      0       0       0 1507680       0 
> table(mergedLogDataSet$protocol[mergedLogDataSet$dst.ip=="192.168.0.51"])

     -      2   ICMP    TCP    UDP 
     0      0 522720      0   1440 
> table(mergedLogDataSet$protocol[mergedLogDataSet$dst.ip=="192.168.0.51" & mergedLogDataSet$action=="DROP"])

     -      2   ICMP    TCP    UDP 
     0      0 522720      0      0 
```

Fig. 16. Summary of protocols for IP address 192.168.0.51

and development of an intelligent system for future incident response and prevention purposes [19]. **R** is a powerful analysis tool which hosts several CI packages related to artificial neural networks, evolutionary algorithms, fuzzy systems and hybrid intelligent systems for designing intelligent systems. Additionally, the use of all these CI techniques in **R** is relatively easy as compared to several other analysis packages. Here, a fuzzy inference system is designed using a *sets* package (see Fig. 17) to predict the risk of attack based on the previous analysis. This is a quite simple design accomplished in only two stages as shown in Figs. 18 and 19.

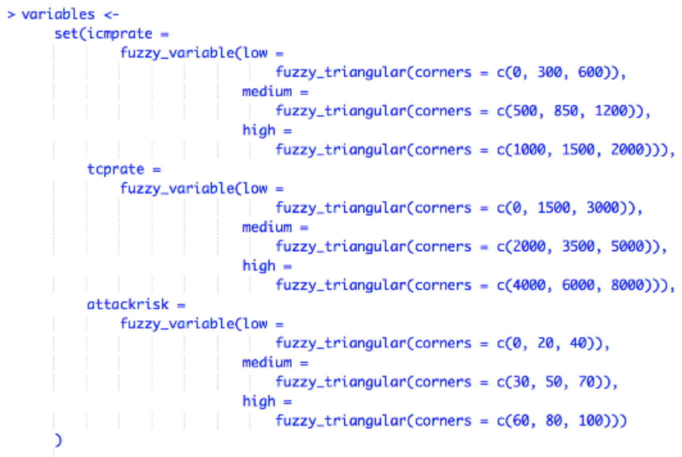
Previous analyses revealed useful information about the traffic data and security issues. However, it does not offer any model to cope with future attacks, and the only way to protect systems from future attacks is still the Firewall rules. Nonetheless, this information can be used to build an intelligent model to monitor future attacks [10], [11], [12], [13]. The detailed analysis of the "mergedLog" file and its dataset unfolded that the rate of ICMP and TCP packets may help system to predict the future risk of the attack in addition to the Firewall rules. Subsequently, for the baselining of this host, the range of ICMP packets (0-2000 packets/second) and TCP packets (0-8000 packets/second) were determined to decide the normal and abnormal traffic conditions. The baseline information is used to design two fuzzy input variables *icmprate* and *tcprate*; its further details can be found in [10], [11], [12], [13]. Based on these two fuzzy input variables, the fuzzy output variable *attackrisk* is determined which predicts the risk of an attack in percentage (0-100). All the fuzzy input and output variables are divided into three fuzzy range *low, medium* and *high* as shown in Fig. 18. Afterwards, a sample fuzzy rule base (see Fig. 19) is designed for the fuzzy inference system (see Fig. 20) to predict the risk of attack. This system can be employed alongside Firewall rules to predict the possibility and level of an attack which is not possible in Windows Firewall; its further details can be found in [10], [11], [12], [13].



```
> install.packages("sets")
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0  0  563k    0     0    0
0      0      0 --:--:-- --:--:-- --:--:--     0100  563k  100  563k    0     0   382k      0  0:00:01  0
:00:01 --:--:--  382k

The downloaded binary packages are in
	/var/folders/g3/94x8nbdd77jcl6s4xmkp0j_00000gn/T//RtmpUdTgTu/downloaded_packages
> library(sets)
```

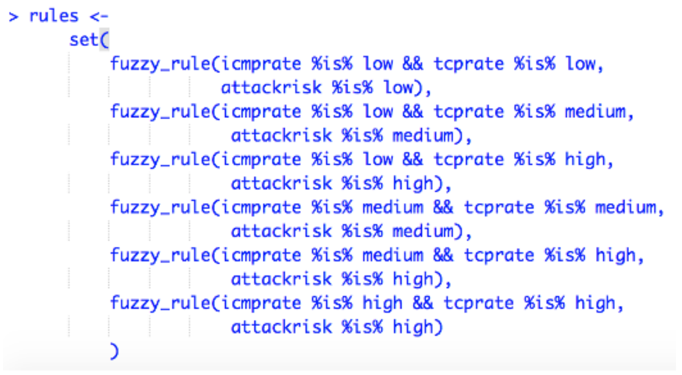
Fig. 17. Installation of Sets package for designing a fuzzy inference system in R

This simple and easy design of the fuzzy intelligent system in **R** to monitor and predict the risk of an attack is only one example of the strength of **R** and its support for CI techniques. The other CI techniques such as artificial neural networks,



```
> variables <-
    set(icmprate =
            fuzzy_variable(low =
                                fuzzy_triangular(corners = c(0, 300, 600)),
                            medium =
                                fuzzy_triangular(corners = c(500, 850, 1200)),
                            high =
                                fuzzy_triangular(corners = c(1000, 1500, 2000))),
        tcprate =
            fuzzy_variable(low =
                                fuzzy_triangular(corners = c(0, 1500, 3000)),
                            medium =
                                fuzzy_triangular(corners = c(2000, 3500, 5000)),
                            high =
                                fuzzy_triangular(corners = c(4000, 6000, 8000))),
        attackrisk =
            fuzzy_variable(low =
                                fuzzy_triangular(corners = c(0, 20, 40)),
                            medium =
                                fuzzy_triangular(corners = c(30, 50, 70)),
                            high =
                                fuzzy_triangular(corners = c(60, 80, 100)))
    )
```

Fig. 18. Defining linguistic fuzzy variables in R



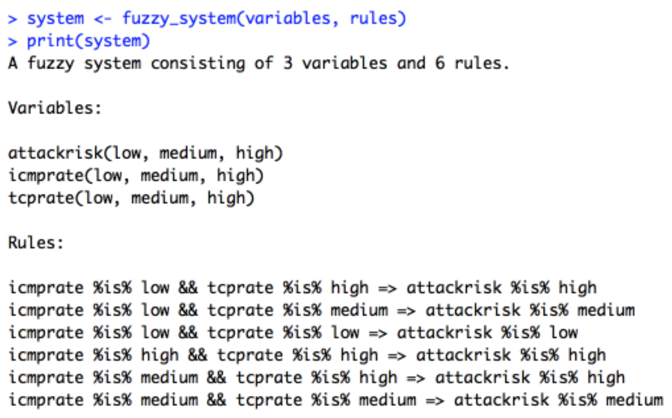
```
> rules <-
    set(
        fuzzy_rule(icmprate %is% low && tcprate %is% low,
                    attackrisk %is% low),
        fuzzy_rule(icmprate %is% low && tcprate %is% medium,
                    attackrisk %is% medium),
        fuzzy_rule(icmprate %is% low && tcprate %is% high,
                    attackrisk %is% high),
        fuzzy_rule(icmprate %is% medium && tcprate %is% medium,
                    attackrisk %is% medium),
        fuzzy_rule(icmprate %is% medium && tcprate %is% high,
                    attackrisk %is% high),
        fuzzy_rule(icmprate %is% high && tcprate %is% high,
                    attackrisk %is% high)
        )
```

Fig. 19. Designing fuzzy rule base in R

evolutionary algorithms and hybrid intelligent systems can also be used in the same way to design various intelligent systems. Additionally, the baseline analysis and range of parameters can be adjusted and manipulated depending on the requirement of a particular host/network.

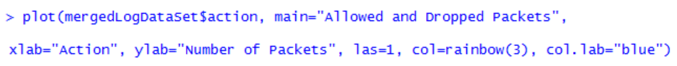
E. Visual Interpretation of Security Analysis Findings using Graphs in ***R***

The final step of this security analysis approach is to present the findings in simple readable and visualised format. **R** is a very powerful tool for data visualisation due to many external packages such as lattice, ggplot2, vcd or hexbin for the enhanced graphics presentation of information [20]. In this security analysis, some of the main findings are presented using the simple built-in graph function "plot", however, the advanced package "ggplot2" can also be used for more informative and appealing presentation. Fig. 21 expresses the plot command and Fig. 22 depicts its resultant information about all allowed, dropped and lost packets, which inform us that the desktop allowed 73% of packets as compared to the dropped and lost 27% of packets. Fig. 23 expresses the plot command to draw source IP addresses and their corresponding actions. Fig. 24 shows the resultant graph of the plot command for how many packets were allowed or dropped from the particular source IP address. The red and green colours shows allowed and dropped packets repectively for that source IP address.



```
> system <- fuzzy_system(variables, rules)
> print(system)
A fuzzy system consisting of 3 variables and 6 rules.

Variables:

attackrisk(low, medium, high)
icmprate(low, medium, high)
tcprate(low, medium, high)

Rules:

icmprate %is% low && tcprate %is% high => attackrisk %is% high
icmprate %is% low && tcprate %is% medium => attackrisk %is% medium
icmprate %is% low && tcprate %is% low => attackrisk %is% low
icmprate %is% high && tcprate %is% high => attackrisk %is% high
icmprate %is% medium && tcprate %is% high => attackrisk %is% high
icmprate %is% medium && tcprate %is% medium => attackrisk %is% medium
```

Fig. 20. Resultant fuzzy inference system in R



```
> plot(mergedLogDataSet$action, main="Allowed and Dropped Packets",
 xlab="Action", ylab="Number of Packets", las=1, col=rainbow(3), col.lab="blue")
```

Fig. 21. Plot command to draw allowed and dropped packets

Similarly, Fig. 25 expresses the plot command to draw destination IP addresses and their corresponding actions. Fig. 26 shows the resultant graph of the plot command for how many packets were allowed or dropped for the particular destination IP address. The red colour shows allowed packets and cyan colour shows dropped packets for that destination IP address. In all security analyses, the nature of the analysis and its interpretations are determined by the user/analyst.

IV. Big Data Scalability for Desktop Users

Any big data analysis approach for desktop users should be able to cope with the increasing volume of data and its effective processing. Today's desktops consist of multi-core processors and increased memory. Therefore, a big data analysis approach should optimise the use of these two resources: processor and memory. **R** is employed in the proposed approach and it can achieve this goal; however, it requires the support of additional packages to make the optimised used of processor and memory

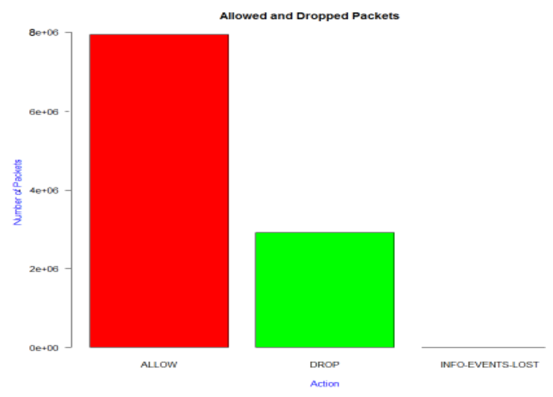
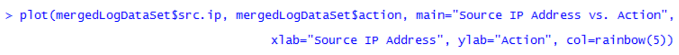


Fig. 22. Illustration of allowed and dropped packets



```
> plot(mergedLogDataSet$src.ip, mergedLogDataSet$action, main="Source IP Address vs. Action",
                                     xlab="Source IP Address", ylab="Action", col=rainbow(5))
```

Fig. 23. Plot command to draw source IP address and corresponding actions

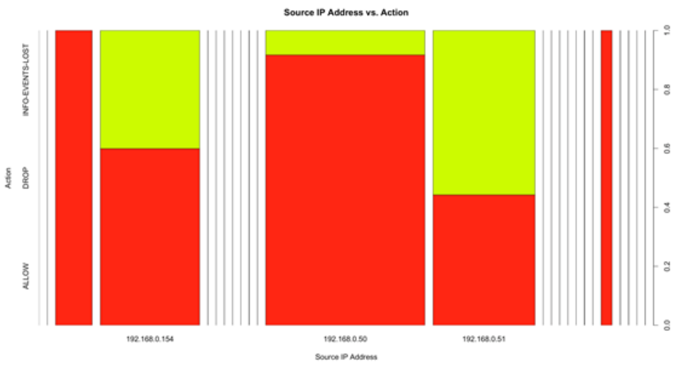
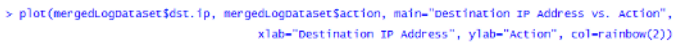


Fig. 24. Illustration of source IP address and corresponding actions



```
> plot(mergedLogDataSet$dst.ip, mergedLogDataSet$action, main="Destination IP Address vs. Action",
                         xlab="Destination IP Address", ylab="Action", col=rainbow(2))
```

Fig. 25. Plot command to draw destination IP address and corresponding actions

[21], [22].

Since its inception, **R** was designed to use only a single thread (processor) at a time. Today, **R** operates the same way unless linked with multi-core/multi-threaded libraries [21]. The multi-core machines of today offer parallel processing power, therefore, to make use of multiple cores, **R** requires the support of add-on packages related to High-Performance and Parallel Computing (HPPC) [23]. There are several packages available for parallel processing in **R** such as parallel, multicore, snow, snowfall, Rmpi, pbdMPI, Rborist, h2o, randomForestSRC, Rdsm, Rhpc. Package “parallel” is built on packages “multicore” and “snow” and provides replacements for most of the functionality of these packages [24]. Package “parallel” handles running much larger chunks of computations in parallel. A typical example is to evaluate the same **R** function on many different sets of data. For Windows desktop users, Microsoft **R** Open includes multi-threaded math libraries to improve the performance of **R** and also works on all OS Windows/Unix/Mac [25]. These libraries make it possible for several common **R** operations, such as matrix multiply/inverse, matrix decomposition, and some higher-level matrix operations, to compute in parallel and use all of the processing power available to reduce computation times [26].

Large datasets also require substantial memory. If the file

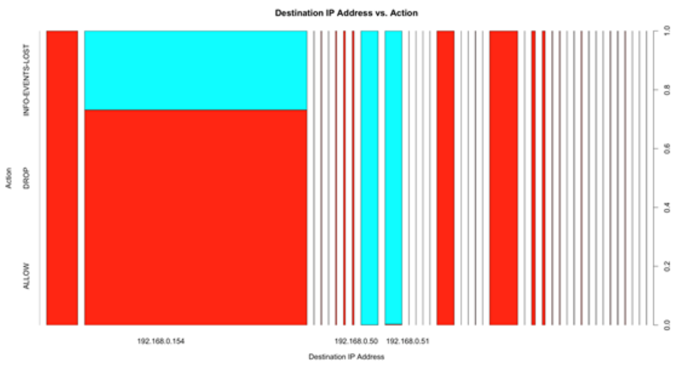


Fig. 26. Illustration of destination IP address and corresponding actions

size is quite large as compared to the existing memory of the system, then "ff" package can be used to perform effective and fast data processing. The "ff" package provides data structures that are stored on disk but behave as if they were in RAM by transparently mapping only a section (pages) in main memory, the effective virtual memory consumption per "ff" object [27]. Another solution to the handling of increasing volume of data is the "big" package family that consists of several packages for performing tasks on large datasets such as bigmemory [28], biganalytics, bigtabulate, synchronicity and bigalgebra [22].

V. Conclusion

This paper has presented an intuitive and inexpensive big data security analysis approach using Computational Intelligent (CI) techniques for Windows desktop users. It is based on the combination of Windows batch script, EmEditor (which can be replaced with any powerful editor) and **R**. This security analysis approach was carried out on a real dataset of $1,006,889,160$ bytes (1.01 GB) with more than 10 million observations, which were collected in the Windows Firewall log file "pfirewall.log" and integrated into the "mergedLog" file over the period of 30 days. This desktop-oriented security analysis deduced the security status of the desktop, and sources and causes of the security breaches successfully. Based on the analyses results, a fuzzy inference system was designed to predict the risk of attack and protect the desktop. This security analysis approach and its successful implementation on the modest desktop configuration show the potential of the proposed approach. However, this particular implementation was limited to the simulated data based on certain firewall rules, few protocols and IP addresses; it would be important to extend rules and areas of investigations, and collect external traffic for making this approach as a generalised security analysis approach.

References

[1] W3schools.com. (2016) OS platform statistics and trends. [Online]. Available: http://www.w3schools.com/browsers/browsers_os.asp

[2] H. Carvey, *Windows Forensic Analysis Toolkit: Advanced Analysis Techniques for Windows 8*. Elsevier, 2014.

[3] A. Cavoukian and J. Jonas, *Privacy by design in the age of big data*. Information and Privacy Commissioner of Ontario, Canada, 2012.

[4] B. Oancea and R. M. Dragoescu, "Integrating R and hadoop for Big Data Analysis," *arXiv preprint arXiv:1407.4908*, 2014.

[5] Cran.r-project.org. (2015) The comprehensive R archive network. [Online]. Available: https://cran.r-project.org/

[6] Microsoft.com. (2009) Overview of windows firewall with advanced security. [Online]. Available: https://technet.microsoft.com/library/6ff0e320-0369-496a-8f1f-0b7224c7f857.aspx

[7] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[8] N. Naik, R. Diao, C. Quek, and Q. Shen, "Towards dynamic fuzzy rule interpolation," in *IEEE International Conference on Fuzzy Systems*, 2013, pp. 1–7.

[9] N. Naik, R. Diao, and Q. Shen, "Genetic algorithm-aided dynamic fuzzy rule interpolation," in *IEEE International Conference on Fuzzy Systems*, 2014, pp. 2198–2205.

[10] N. Naik, "Fuzzy inference based intrusion detection system: FI-Snort," in *IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2015, pp. 2062–2067.

[11] N. Naik and P. Jenkins, "Fuzzy reasoning based windows firewall for preventing denial of service attack," in *IEEE International Conference on Fuzzy Systems*, 2016.

[12] N. Naik, R. Diao, and Q. Shen, "Application of dynamic fuzzy rule interpolation for intrusion detection: D-FRI-Snort," in *IEEE International Conference on Fuzzy Systems*, 2016.

[13] N. Naik and P. Jenkins, "Enhancing windows firewall security using fuzzy reasoning," in *IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2016, pp. 263–269.

[14] E. H. Mamdani and S. Assilina, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[15] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 116–132, 1985.

[16] D. Fisher, R. DeLine, M. Czerwinski, and S. Drucker, "Interactions with big data analytics," *Interactions*, vol. 19, no. 3, pp. 50–59, 2012.

[17] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, pp. 1–21, 2015.

[18] Emeditor.com. (2015) Text editor for windows. [Online]. Available: https://www.emeditor.com/

[19] J. G. Alfaro, N. Boulahia-Cuppens, and F. Cuppens, "Complete analysis of configuration rules to guarantee reliable network security policies," *International Journal of Information Security*, vol. 7, no. 2, pp. 103–122, 2008.

[20] W. Cho, Y. Lim, H. Lee, M. K. Varma, M. Lee, and E. Choi, "Big data analysis with interactive visualization using R packages," in *Proceedings of the 2014 International Conference on Big Data Science and Computing*. ACM, 2014, p. 18.

[21] R. R. Rosario. (2010, July 27) Taking R to the limit, Part I: Parallelization. [Online]. Available: http://www.bytemining.com/wp-content/uploads/2010/07/r_hpc.pdf

[22] ——. (2010, August 17) Taking R to the limit, Part II: Working with large datasets. [Online]. Available: http://www.bytemining.com/wp-content/uploads/2010/07/r_hpc.pdf

[23] D. Eddelbuettel. (2016, October 10) CRAN Task View: High-Performance and Parallel Computing with R. [Online]. Available: https://cran.r-project.org/web/packages/bigmemory/index.html

[24] R-core. (2015, December 4) Package 'parallel'. [Online]. Available: https://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf

[25] Mran.microsoft.com. (2016, September 1) Microsoft R Open: The Enhanced R Distribution. [Online]. Available: https://mran.microsoft.com/open/

[26] ——. (2016, September 1) About Microsoft R Open: The Enhanced R Distribution. [Online]. Available: https://mran.revolutionanalytics.com/rro/

[27] Cran.r-project.org. (2014, April 9) ff: memory-efficient storage of large data on disk and fast access functions. [Online]. Available: https://cran.r-project.org/web/packages/ff/index.html

[28] M. J. Kane, J. W. Emerson, P. Haverty, and C. Determan. (2016, March 28) bigmemory: Manage massive matrices with shared memory and memory-mapped files. [Online]. Available: https://cran.r-project.org/web/packages/bigmemory/index.html