Security Visualization: Detecting Denial of Service

Glen Hawthorne, Ying He(✉), Leandros Maglaras, and Helge Janicke

De Montfort University, Leicester LE1 9BH, UK
grhawthorne@googlemail.com,
{ying.he,leandros.maglaras,heljanic}@dmu.ac.uk

Abstract. Denial Of Service attacks are notorious attack methods used to target servers of IT systems and Industrial Control Systems to prevent them from working or to reduce efficiency, hence decreasing user experience. Visualization is the method of taking data, processing and displaying data in an easy to view format. Visualization could be used to identify Denial Of Service attacks by monitoring the data sent to clients and being displayed to the users. Manipulating the type of data shown and the format it is shown in can help users spot potential attacks by seeing outliers in the data sets. This research develops novel software that can run on an web server. It processes the web access logs, displays the data to users and identify potential attacks in access logs. The software has been tested, with the majority of tests passing. Further development of the project is discussed and the main areas for development are also explored.

Keywords: Denial of service attack · Security visualization · Web logs · Intrusion detection

1 Introduction

Denial Of Service and Distributed Denial Of Service attacks have been proven to be constant threats as over two thousand attacks are launched daily and can be purchased via the black market, allowing those who lack technical skills to launch attacks [17]. According to previous research, speed is an important feature and the software will need to match reading 70000 records a second from a web log [18]. The data need to be presented in a clear way that allows quick reading by the user and operate independently of web server applications. Tools are available for monitoring web servers but can only operate with specific web servers. This paper reviews the most popular web servers (Apache, nginx, Microsoft IIS and Google Server), and identifies the format they utilize to save the web logs that allow software to process them.

In the three main control systems of a Critical Infrastructure, the SCADA is the central nerve system that constantly gathers the latest status from remote units. Many researchers have applied Internet technologies in order to improve

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2017
L.A. Maglaras et al. (Eds.): INISCOM 2016, LNICST 188, pp. 39–49, 2017.
DOI: 10.1007/978-3-319-52569-3_4

certain functions in a SCADA system. Authors in [19] present a web-based SCADA display system which is implemented based on the client/server architecture that can control the operation of a substation on the server side. Authors in [20] present a Web-based power quality monitoring system that is allowing users to operate the system through the browser. SCADA systems are vulnerable to many attacks [21,24] and new Intrusion Detection Systems that can monitor both the network traffic [23] or the web server activity [22] are of great need.

In order to help users identify when their server is targeted by a Denial Of Service attack, this article presents a novel software to analyze web server access logs. Using the data extracted from the access logs, graphs are created and visually displays users different loads on the web server. The developed application is also capable of detecting possible HTTP Flood denial (and distributed) of service attacks.

This paper is structured as follows. Section 2 reviews related work of Denial Of Service attacks; Sect. 3 presents existing products, web server applications and logs; Sect. 4 introduces the software developed that can extract information from the logs to detect attacks and visualize information to the users and Sect. 5 summarizes the paper and discusses future work.

2 Related Work

Denial Of Service attacks can render online resources inaccessible/unavailable to users [2]. They are not designed to breach security but can be used as a distraction technique for other malicious practices. Being a high profile form of attack, Denial Of Service attack is a popular method used by hacktivists, cyber vandals and other similar groups. The attacks are capable of lasting for extended periods of time and can be damaging to the victim organisation in terms of monetary loss, loss of customer trust and damage to reputation.

Denial Of Service attacks are attacks sent from a single host, however, when an attack is sent from multiple hosts it is classed as a Distributed Denial Of Service attack [1]. Distributed Denial Of Service attacks are usually launched via the use of a botnet.

UDP Flood Denial Of Service attacks use IP packets that contain UDP datagrams [3]. When the targeted machine receives the packets it checks for programs associated with the received datagrams. When no associated program can be found the victim machine sends a destination unreachable response. As more and more UDP packets are sent to the target machine and are responded too, the victim will eventually become overwhelmed and unable to respond to user requests.

SYN Flood is a form of Denial Of Service attack that exploits vulnerabilities in the TCP three way handshake protocol and is designed to abuse resources on the targeted machine [4]. The attacking machine(s) will send TCP connection requests faster than the victim is capable of processing them. The attack works via the attacker sending repeated SYN packets to all ports on the targeted server, the server in turn replying to each sent packet with a SYN ACK packet while the attacking machine is not responding to the SYN ACK packets as would normally happen. The server then waits for acknowledgment (for the SYN ACK) and while waiting for this response the connection cannot be closed. Before the connection times out more SYN packets are received resulting in an increasing amount of open connections. This leads to the server maxing out on the amount of connections it can handle and legitimate users being denied access to resources. SYN packets can be sent with fake IP addresses by the attacker, making the victim server respond to fake addresses.

HTTP Flood attack is another form of Denial Of Service attack that often utilize botnets [5]. For a HTTP Flood attack to be successfully executed, knowledge of the target is required, meaning that each attack must be specifically designed, making it hard to detect. This form of attack utilizes GET and POST methods. The POST method is the most effective for this attack as the parameters passed require dynamic processing on the server side absorbing greater resources from the victim. The GET method can be used to get static content from the target server. POST is a more effective form of HTTP Flood due to the extra processing that is required on server side, however, GET is a more scalable in distributed denial of service attacks.

The Ping of Death is a Denial Of Service attack that allows malicious users to send IP packets greater than the 65532 byte limit [6]. Fragmentation is a feature offered with TCP packets that allows the packets to be broken down into smaller segments. It is possible to send packets that exceed the 65536 byte limit with the use of fragmentation. When operating systems receive a packet that exceed the limit, they would freeze, crash or reboot. It also allows for attackers to remain anonymous as the identify of the attacker can be easily falsified. The only detail an attacker would need to know about the victim machine is the IP address.

3 Existing Products

Logstalgia is marketed as a website traffic visualization tool that is capable of watching live as well as replaying requests made to a web server, it is designed to be in the style of the classic computer game Pong [7]. The log formats supported by Logstalgia are NCSA Common Log Format, NCSA Common Log Format with Virtual Host, NCSA Extended/Combined Log Format and NCSA Extended/Combined Log Format with Virtual Host. However, the software is dependent upon OpenGL and is recommended to run on a workstation rather than the web server. The traffic is represented via coloured blocks traveling across the screen, from the clients on the left to the web server on the right of the screen.

Nginx is a popular web server. The nginx Plus version provides the user with a real time monitoring interface that provides information on key load and performance issues [8]. The monitoring interface can provide a range of information on the server and running details. It provides administrative information on the server including the version number running and its IP address. It also shows the amount of connections to the server, number of requests, detailed information on server zones and up-streams as well as the amount of traffic going through the TCP zones.

The Webalizer is a web server log analyzer that produces reports in HTML format [9]. It is fast and claims to be able to process roughly 70000 records per second on a 1.6 GHZ laptop [9]. It supports a wide range of web server log file layouts which includes Common Log File Format, several variations of NCSA Combined Logfile format, wu-ftp/proftpd xferf format logs, Squid proxy server native format and W3C Extended log formats. The Webalizer is capable of analyzing compressed web logs without the need of uncompressing them. When analyzing log files there is no limit to the size of the file being analyzed and is also capable of analyzing partial logs. The Webalizer is capable of generating reports via command line, supports IPv4 and IPv6 addressing and has built in geolocation services.

Imperva Incapsula is a piece of software that provides Distributed Denial Of Service attack mitigation and aims to protect against layer 3, 4 and 7 Denial Of Service attacks [10]. It can be installed in less than five minutes, offering automatic detection and activation with an extremely low rate of false positives. Running in real time it is capable of blocking multi-gigabit attacks and can also prevent SQL injection and cross site scripting attack. The claim is that it will not slow down the site running it, but make the site run faster and more efficiently, using less resources by offering various acceleration techniques. No server down time is required during installation of the software.

3.1 Web Server Applications

There are numerous web server applications and they can run on a variety of operating systems. Table 1 shows that Apache is currently the most widely used web server application as of January 2015. Table 2 shows the most popular (for the top one million domains as of 2012) operating system to run when a hosting web server is Linux.

Table 1. Most popular web servers

Web server application	Percentage of market share [12]
Apache	39.74 %
Microsoft	27.52 %
nginx	14.47 %
Google	2.30 %
Other	15.83 %

3.2 Web Server Logs

There are numerous standardized ways in which web servers can store their logs, each variety having different information and different way of storing it. The

Table 2. Operating systems running web server applications

Operating systems	Percentage of operating systems running web servers [13]
Linux	46.30 %
CentOS	16.60 %
RedHat	10.90 %
Ubuntu	9.00 %
Debian	7.67 %
ebian	7.67 %
Fedora	3.51 %
FreeBSD	2.06 %
Windows	1.27 %
SUSE	1.05 %

Table 3. Explanation of fields in NCSA common log file format

Format	Meaning [14]
remotehost	The remote host name of IP address of the client
rfc931	Remote log name of the client
authuser	The name the client is authorised with
date	Date and time the request was made
request	Request line as it came from the client
status	The HTTP status code returned to the user
bytes	The content length of the document transferred

NCSA Common Log File Format is a standardized way to layout web logs. It is outlined below and explained in Table 3. In some web servers, such as Apache, a hyphen is used in the log-in indicating a field is not available for the log [11].

The W3C Extended Log Format is another standardized way in which a web server can store its logs, it was designed to be an improved format of the Common Log File Format for storing web logs [15]. It allows for additional information to be included (compared to the Common Log Format) in the logs or selected information to be omitted from the logs. Table 4 below outlines and explains what each of the possible fields in the W3C Extended Log Format is used for.

The W3C Extended Log Format also provides additional information at the start of the log [15], It holds the version of the W3C Extended Log Format being used, the fields recorded in the log, the software that generated the logs, the date and time the log was started, the date and time the log ended, the date and time in which the entry was added and a comment section [15].

The Combined Log Format offers the same information as the common log format, plus additional fields [16]. The extra data offered includes: the name of the site that the user was on if they followed a link, information about the user

Table 4. Explanation of fields in WC extended log format

Field	Explanation
Date	Date at which transaction was completed
Time	Time at which transaction was completed
Time-taken	Time taken for transaction to compete, measured in seconds
Bytes	Amount of bytes transferred
Cached	Whether a cache hit occurred
IP	The IP address and port
DNS	The DNS name
Status	The status code returned
Comment	The comment returned with the status code
Method	The method
URI	The URI
URI-stem	Stem part of the URI
Count	The number of entries for the listed data
Time-from	Time that the sampling began
Time-to	Time that the sampling ended
Interval	Length of time that sampling occurred for, recorded in seconds

i.e. browser and browser version and operating system used as well as information on cookies that we sent by the HTTP server [16].

3.3 Web Servers and Web Logs

Having discovered the most popular web server applications used and some common standardized web log layouts, the web log layout that web severs offer, needs to be researched also.

Google allows members of the public to host their websites on Google servers. It is thought that these servers run a custom web server built and maintained by Google. Due to the proprietary nature of this web server it will not be possible to discover how Google layout their web logs.

The Apache web server supports Common Log Format, Combined Log Format and Conditional Logs. The nginx web server supports Combined Log Format and Modified/custom variation of the Combined Log Format. Microsoft IIS supports W3C Extended Log File Format, W3C Centralised Logging, NCSA Common Log File Format, IIS Log File Format, ODBC Logging, Centralised Binary Logging and HTTP.sys Error Log Files.

4 Proposed Software Application

The main functionality of this software product is to extract information from the logs to detect an attack and manipulate the data from the access logs to present information to the user that is useful and can visually identify attacks on the server.

4.1 Software Introduction

The software allows users to enter the path to web access logs, view the data contained in the logs, filter the data to view and receive and alert to address that could contain denial of service attacks. It was decided to investigate the data available in the logs after an attack on a web server as the access logs are usually overlooked by some products on the market and could contain useful information. The software processes NCSA Common Log File Format and Combined Log Format web logs.

Figure 1 displays an example of the graph that is used to show the total load on the web server. Figure 2 shows the class diagram for the main classes used by the produced software. The diagram shows how classes interact. By examining the diagram it is obvious that the program abides with Model View Controller (MVC) [26] principles. More classes were used in the construction of the software, however, these are items such as file readers and data structures that would crowd the diagram and are not necessary to show the flow of the program. Figure 3 displays the point chart of addresses. Each address has a different color, allowing for easy identification of addresses.

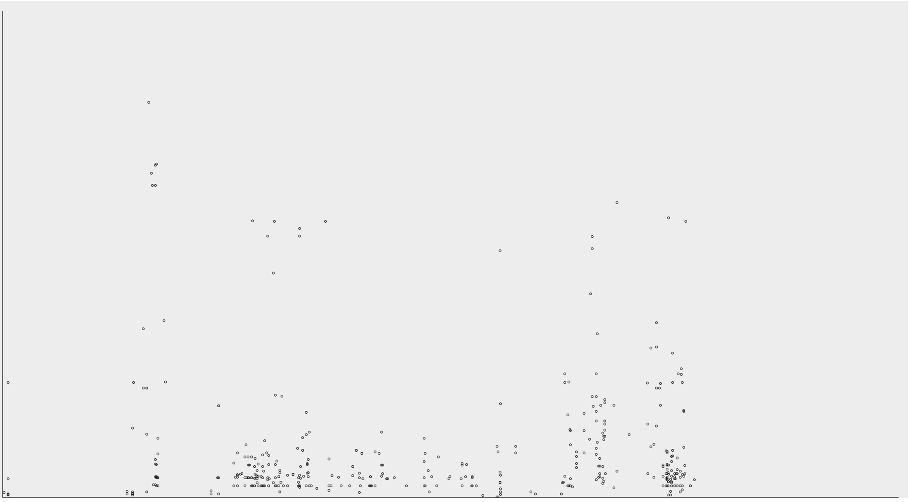

Fig. 1. The total load on the web server

The software has been designed to help build learned actions during repeated use, allowing experienced users to quickly navigate the menus effectively. As explained by [25], once actions have been completed repetitively they start to become learned behaviour and can be performed without the need to think about how to achieve the task.

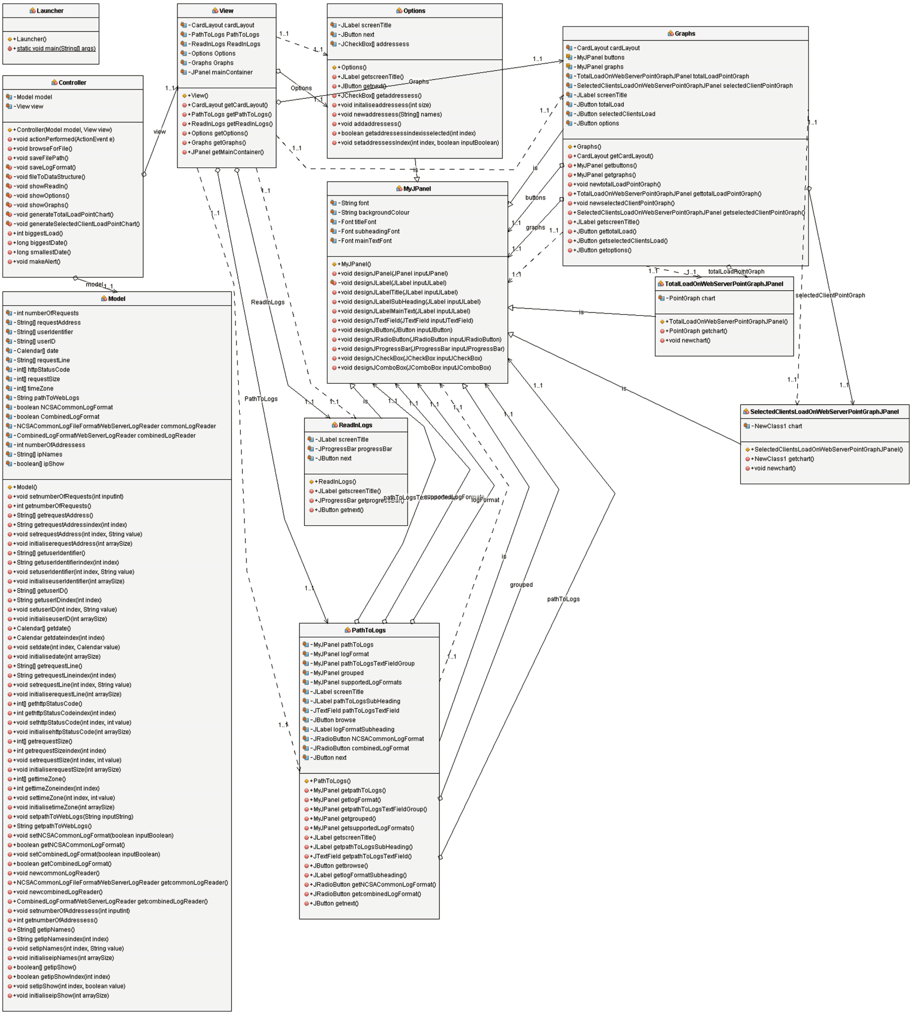
Launcher
+Launcher()
+static void main(String[] args)
Controller
-Model model
-View view
+Controller(Model model, View view)
+void actionPerformed(ActionEvent e)
+void browseForFile()
+void saveFilePath()
-void saveLogFormat()
-void fileToDataStructure()
-void showReadIn()
-void showOptions()
-void showGraphs()
-void generateTotalLoadPointChart()
-void generateSelectedClientLoadPointChart()
+int biggestLoad()
+long biggestDate()
+long smallestDate()
+void makeAlert()
View
-CardLayout cardLayout
-PathToLogs PathToLogs
-ReadInLogs ReadInLogs
-Options Options
-Graphs Graphs
-JPanel mainContainer
+View()
+CardLayout getCardLayout()
+PathToLogs getPathToLogs()
+ReadInLogs getReadInLogs()
+Options getOptions()
+Graphs getGraphs()
+JPanel getMainContainer()
Options
-JLabel screenTitle
-JButton next
-JCheckBox[] addressess
+Options()
+JLabel getscreenTitle()
+JButton getnext()
+JCheckBox[] getaddressess()
+void initialiseaddressess(int size)
+void newaddressess(String[] names)
+void addaddressess()
+boolean getaddressessindexisselected(int index)
+void setaddressessIndex(int index, boolean inputBoolean)
Graphs
-CardLayout cardLayout
-MyJPanel buttons
-MyJPanel graphs
-TotalLoadOnWebServerPointGraphJPanel totalLoadPointGraph
-SelectedClientsLoadOnWebServerPointGraphJPanel selectedClientPointGraph
-JLabel screenTitle
-JButton totalLoad
-JButton selectedClientsLoad
-JButton options
+Graphs()
+CardLayout getCardLayout()
+MyJPanel getbuttons()
+MyJPanel getgraphs()
+void newtotalLoadPointGraph()
+TotalLoadOnWebServerPointGraphJPanel gettotalLoadPointGraph()
+void newselectedClientPointGraph()
+SelectedClientsLoadOnWebServerPointGraphJPanel getselectedClientPointGraph()
+JLabel getscreenTitle()
+JButton gettotalLoad()
+JButton getselectedClientsLoad()
+JButton getoptions()
MyJPanel
-String font
-String backgroundColour
-Font titleFont
-Font subheadingFont
-Font mainTextFont
+MyJPanel()
+void designJPanel(JPanel inputJPanel)
-void designJLabel(JLabel inputJLabel)
+void designJLabelTitle(JLabel inputJLabel)
+void designJLabelSubHeading(JLabel inputJLabel)
+void designJLabelMainText(JLabel inputJLabel)
+void designJTextField(JTextField inputJTextField)
+void designJButton(JButton inputJButton)
+void designJRadioButton(JRadioButton inputJRadioButton)
+void designJProgressBar(JProgressBar inputJProgressBar)
+void designJCheckBox(JCheckBox inputJCheckBox)
+void designJComboBox(JComboBox inputJComboBox)
TotalLoadOnWebServerPointGraphJPanel
-PointGraph chart
+TotalLoadOnWebServerPointGraphJPanel()
+PointGraph getchart()
+void newchart()
SelectedClientsLoadOnWebServerPointGraphJPanel
-NewClass1 chart
+SelectedClientsLoadOnWebServerPointGraphJPanel()
+NewClass1 getchart()
+void newchart()
ReadInLogs
-JLabel screenTitle
-JProgressBar progressBar
-JButton next
+ReadInLogs()
+JLabel getscreenTitle()
+JProgressBar getprogressBar()
+JButton getnext()
Model
-int numberOfRequests
-String[] requestAddress
-String[] userIdentifier
-String[] userID
-Calendar[] date
-String[] requestLine
-int[] httpStatusCode
-int[] requestSize
-int[] timeZone
-String pathToWebLogs
-boolean NCSACommonLogFormat
-boolean CombinedLogFormat
-NCSACommonLogFileFormatWebServerLogReader commonLogReader
-CombinedLogFormatWebServerLogReader combinedLogReader
-int numberOfAddresses
-String[] ipNames
-boolean[] ipShow
+Model()
+void setnumberOfRequests(int inputInt)
+int getnumberOfRequests()
+String[] getrequestAddress()
+String getrequestAddressindex(int index)
+void setrequestAddress(int index, String value)
+void initialiserequestAddress(int arraySize)
+String[] getuserIdentifier()
+String getuserIdentifierindex(int index)
+void setuserIdentifier(int index, String value)
+void initialiseuserIdentifier(int arraySize)
+String[] getuserID()
+String getuserIDindex(int index)
+void setuserID(int index, String value)
+void initialiseuserID(int arraySize)
+Calendar[] getdate()
+Calendar getdateindex(int index)
+void setdate(int index, Calendar value)
+void initialisedate(int arraySize)
+String[] getrequestLine()
+String getrequestLineindex(int index)
+void setrequestLine(int index, String value)
+void initialiserequestLine(int arraySize)
+int[] gethttpStatusCode()
+int gethttpStatusCodeindex(int index)
+void sethttpStatusCode(int index, int value)
+void initialisehttpStatusCode(int arraySize)
+int[] getrequestSize()
+int getrequestSizeindex(int index)
+void setrequestSize(int index, int value)
+void initialiserequestSize(int arraySize)
+int[] gettimeZone()
+int gettimeZoneindex(int index)
+void settimeZone(int index, int value)
+void initialisetimeZone(int arraySize)
+void setpathToWebLogs(String inputString)
+String getpathToWebLogs()
+void setNCSACommonLogFormat(boolean inputBoolean)
+boolean getNCSACommonLogFormat()
+void setCombinedLogFormat(boolean inputBoolean)
+boolean getCombinedLogFormat()
+void newcommonLogReader()
+NCSACommonLogFileFormatWebServerLogReader getcommonLogReader()
+void newcombinedLogReader()
+CombinedLogFormatWebServerLogReader getcombinedLogReader()
+void setnumberOfAddresses(int inputInt)
+int getnumberOfAddresses()
+String[] getipNames()
+String getipNamesindex(int index)
+void setipNames(int index, String value)
+void initialiseipNames(int arraySize)
+boolean[] getipShow()
+boolean getipShowIndex(int index)
+void setipShow(int index, boolean value)
+void initialiseipShow(int arraySize)
PathToLogs
-MyJPanel pathToLogs
-MyJPanel logFormat
-MyJPanel pathToLogsTextFieldGroup
-MyJPanel grouped
-MyJPanel supportedLogFormats
-JLabel screenTitle
-JLabel pathToLogsSubHeading
-JTextField pathToLogsTextField
-JButton browse
-JLabel logFormatSubheading
-JRadioButton NCSACommonLogFormat
-JRadioButton combinedLogFormat
-JButton next
+PathToLogs()
+MyJPanel getpathToLogs()
+MyJPanel getlogFormat()
+MyJPanel getpathToLogsTextFieldGroup()
+MyJPanel getgrouped()
+MyJPanel getsupportedLogFormats()
+JLabel getscreenTitle()
+JLabel getpathToLogsSubHeading()
+JTextField getpathToLogsTextField()
+JButton getbrowse()
+JLabel getlogFormatSubheading()
+JRadioButton getNCSACommonLogFormat()
+JRadioButton getcombinedLogFormat()
+JButton getnext()
view
model
Options
Graphs
ReadInLogs
PathToLogs
is
buttons
graphs
totalLoadPointGraph
selectedClientPointGraph
pathToLogsTextFieldGroup
supportedLogFormats
logFormat
grouped
pathToLogs
1..1

Fig. 2. The project files class diagram

4.2 Review of Alert Mechanism

The alert mechanism runs after the web server access logs have been processed and saved into the memory. The total load of the server is calculated by taking the sum of all pages sent by the web server. This value is divided by the number of unique addresses found in the provided web access log in order to calculate the average load per address. Although this may seem to be a primitive way of detecting an attack, HTTP Flood attacks swamp the server with too many requests, which means that any address that initiates an HTTP Flood attack produces a load on the web server that is way above average.

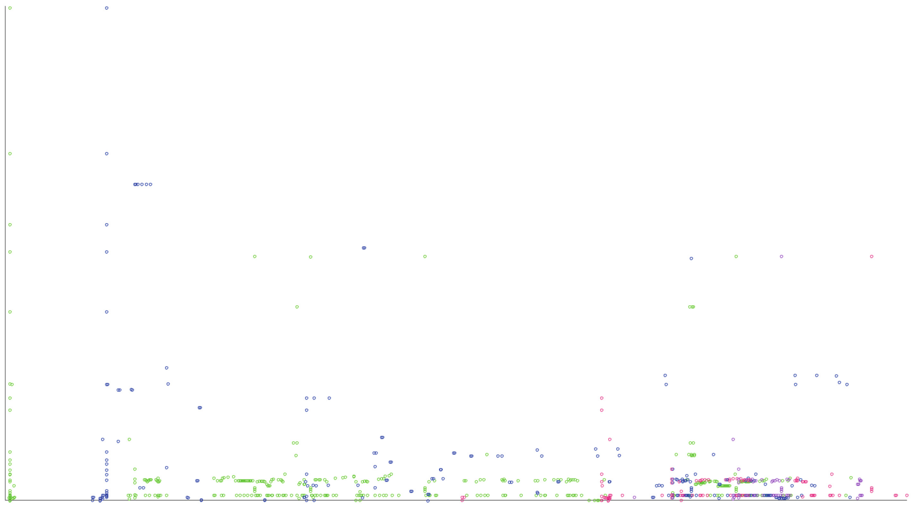

Fig. 3. The point chart of addresses

The addresses that are identified as potential Denial Of Service attack sources, are displayed in an application modal pop up. The pop up is effective for delivering the information for the application in its present format, however, for a log file with a small number of addresses it may prove to be inaccurate. The more addresses there exist in the network, the more effective the attack detection is. In the future when application will be adapted to run in real time, a log file for the application could be used to store the detected attacks. This will allow the user to store the information for future reference and offer other software the choice to access the information found.

4.3 Review of Testing

There were three branches of the testing: Specification Testing, Manual Testing Of Software and Automated Testing. The testing of the software has been positive, meeting the important areas in the specification, behaving as expected during use and passing all automated testing meaning the software produced is robust with a minimum chance of bugs.

The Specification Testing had mixed results during the test. The majority of tests in the Required Specification Test Report were passed. The project was written in Java and is compliant with MVC specification while being platform independent. Files used in the project are documented with JavaDoc. The developed software can process the NCSA Common Log File Format and is capable of reading more than 70000 line from web logs per second. The software is capable of comparing multiple selected clients load on the web server, and all clients load on the web server. HTTP Flood attacks can be automatically detected and visualized via the graphs made. SYN Flood attacks and UDP Flood attacks cannot be detected via the software as they are not saved into the web logs.

5 Summary and Future Work

This paper has explored the problems that Denial Of Service attacks can cause, the products that are currently on the market and the features that they offer. The most popular web logs are examined along with the percentage of operating systems that host web server applications. The format of the most commonly used web logs are examined and the format of web logs that web servers can save in are listed. The article also presents a novel software application that can be used in order to detect UDP Flood, SYN Flood and HTTP Flood attacks on a web server that runs the proposed software. The application can extract information from the logs and use them in order to detect an attack and can use the data from the access logs in order to visualize attack information to the users. The software has been tested, and the results are positive.

The main focus of further development would be to modify the software to enable it to run in real time. This would provide a greater usability feature as it could be used to identify attacks as they are taking place. Different log formats could be supported by the software, allowing compatibility with more set ups and providing the opportunity of greater market share. More graph types could be added to the application, such as the three dimensional graph, providing more ways to visualize the data. Different filtering methods could also be used, such as filtering on the HTTP status code of a request, providing sophisticated mechanisms for filtering out misleading or irrelevant data before creating the graphs.

References

1. Bartholemy, A., Chen, W.: An examination of distributed denial of service attacks. In: 2015 IEEE International Conference on Electro/Information Technology (EIT), pp. 274–279. IEEE (2015)
2. Garber, L.: Denial-of-service attacks rip the Internet. Computer **33**(4), 12–17 (2000)
3. Wan Mohd Ghazali, K., Hassan, R.: Flooding distributed denial of service attacks-a review. J. Comput. Sci. **7**(8), 1218–1223 (2011)
4. Lemon, J.: Resisting SYN flood DoS attacks with a SYN cache. In: BSDCon, vol. 2002, pp. 89–97 (2002)
5. Yatagai, T., Isohara, T., Sasase, I.: Detection of HTTP-GET flood attack based on analysis of page access behavior. In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PacRim 2007, pp. 232–235. IEEE (2007)
6. Kenney, M.: Ping of death. Insecure.org (1996)
7. Logstalgia (2015). http://logstalgia.io/. Accessed 31 Oct 2015
8. Nginx: Logging AND monitoring (2015). https://www.nginx.com/resources/admin-guide/logging-and-monitoring/. Accessed 31 Oct 2015

9. Webalizer: The Webalizer (2014). http://www.webalizer.org/. Accessed 30 Oct 2015
10. Imperva: Why Incapsula? (2015). https://www.incapsula.com/ddos/why-incapsula/. Accessed 13 Nov 2015
11. The Apache Software Foundation: Log files (2015). https://httpd.apache.org/docs/trunk/logs.html#page-header. Accessed 30 Oct 2015
12. Netcraft: January 2015 web server survey (2015). http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html. Accessed 20 Oct 2015
13. SolveDNS statistics (2015). http://www.solvedns.com/statistics/. Accessed 27 Oct 2015
14. World Wide Web Consortium: Logging control In W3C httpd (1995). http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format. Accessed 30 Oct 2015
15. World Wide Web Consortium: Extended log file format (2015). http://www.w3.org/TR/WD-logfile.html. Accessed 31 Oct 2015
16. Ogbuji, U.: Working with web server logs (2009). IBM. http://www.ibm.com/developerworks/library/wa-apachelogs/. Accessed 01 Nov 2015
17. Sauter, M.: LOIC will tear us apart the impact of tool design and media portrayals in the success of activist DDOS attacks. Am. Behav. Sci. **57**(7), 983–1007 (2013)
18. Kenkre, P.S., Pai, A., Colaco, L.: Real time intrusion detection and prevention system. In: Satapathy, S.C., Biswal, B.N., Udgata, S.K., Mandal, J.K. (eds.) Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014. AISC, vol. 327, pp. 405–411. Springer, Heidelberg (2015). doi:10.1007/978-3-319-11933-5_44
19. Qiu, B., Gooi, H.B.: Web-based SCADA display systems (WSDS) for access via Internet. IEEE Trans. Power Syst. **15**(2), 681–686 (2000)
20. Leou, R.-C., Chang, Y.-C., Teng, J.-H.: A web-based power quality monitoring system. In: Power Engineering Society Summer Meeting, vol. 3. IEEE (2001)
21. Maglaras, L.A., Jiang, J.: Intrusion detection in SCADA systems using machine learning techniques. In: Science and Information Conference (SAI). IEEE (2014)
22. Zuech, R., Khoshgoftaar, T.M., Wald, R.: Intrusion detection and big heterogeneous data: a survey. J. Big Data **2**(1), 1–41 (2015)
23. Maglaras, L.A., Jiang, J., Cruz, T.J.: Combining ensemble methods and social network metrics for improving accuracy of OCSVM on intrusion detection in SCADA systems. J. Inf. Secur. Appl., 4 May 2016. ISSN 2214-2126
24. Nicholson, A., Webber, S., Dyer, S., Patel, T., Janicke, H.: SCADA security in the light of cyber-warfare. Comput. Secur. **31**(4), 418–436 (2012)
25. Johnson, J.: Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines. Elsevier, Amsterdam (2013)
26. Syromiatnikov, A., Weyns, D.: A journey through the land of model-view-design patterns. In: 2014 IEEE/IFIP Conference on Software Architecture (WICSA), pp. 21–30, IEEE, April 2014