International Journal of Computer Applications (0975 – 8887)
Volume 105 – No. 14, November 2014

DOS Attack Reduction by using Web Service Filter

Sonali Utsai
Student
MMCOE, Karve Nagar,
Pune - 57

Ram B. Joshi
Guide
H.O.D. of Computer Science Dept.
MMCOE, Karve Nagar,
Pune - 57

ABSTRACT

As Application Denial of Service attacks have rapidly become a commonplace threat for doing business on the Internet - more proof that Web application security is more critical now than ever. Denial of Service attacks can result in significant loss of service, money and reputation for organizations. The paper is proposed to build application layer filters to provide real time detection and mitigation of Daniel of service attack. Web Service filters helps protecting Web Service application and service disruption by removing application level DoS attacks. This can defend critical Web Service resource from attack while relying sophisticated filtering technologies to allow legitimate traffic to continue to flow. We compared the couple of filter designs and how they address the specific Web Service attack types. We briefly discuss the different common DoS attacks, risk associated with them and detail of Web Service filters benefits to the Web Service. Also, we carried out design, implementation and deployment details on a one of commonly used application server.

General Terms

DoS attack, REST, Security, Web Service, Filter etc.

1. INTRODUCTION

According to the W3C, a Web Services primarily supports interoperable machine-to-machine interaction over heterogeneous enterprise networks. Whereas traditional distributed messaging does not provide the capability to connect heterogeneous systems separated by firewalls across organizational and enterprise boundaries seamlessly. Thus Web Services have evolved as a practical, cost-effective solution for uniting information distributed between critical applications over operating system, platform, and language barriers that were previously impossible.

As Web Services become more and more popular, inter-enterprise communications and security is becoming crucial for operating Web Services, while the basic Web Service specifications themselves do not address any security topics. For Web Services a large number of additional specifications like WS-Security, WS-Security Policy, WS-Trust, WS-Secure Conversation etc. are required. However these all standards focus on the aspects of message integrity, confidentiality, user authentication and authorization etc. Very few efforts have been made so far to secure the Web Service server itself and ensure a Web Service’s availability.

In this Paper we present target attack detection by maintaining a little client's requests history and by using Web Service filters. The design and implementation of such filters require special use of filter functionality and placement in order to provide an accurate attack detection solution. This system is for protecting Web Services from Denial-of-Service (DoS) attacks by processing client’s requests before forwarding them to the Server. The paper represents couple of reusable WS (Web Service) filters, designed for implementing known attack detection techniques. Additionally, it provides results from each implemented filter in chain and holds ability to respond to clients request without hitting to actual Web Service. Web Service filters are request message interceptors that can be easily plugged in to the Web service runtime to do additional processing of the inbound messages. The reusability of these components across the services has absolute benefit that the framework brings to service delivery which also improves WS's performance. This mechanism also allows the separation of the most fundamental concerns of application software in Web Services development, effectively abstracting the system service into handlers/filters and leaving the clients and services to focus on business logic.

1.1 Types of Denial of Service

Availability is one of the three main objectives of computer security, along with Confidentiality and integrity. Availability can be defined as the ability to use the information or resource desired. However, this definition of availability skips an important aspect - timeliness. According to the Code of Laws of the United States regarding the definition of information security, availability means ensuring timely and reliable access to and use of information."

Denial of Service is a threat that potentially violates the availability of a resource in a system. A Denial of Service Attack, on the other hand, is an action (or set of actions) executed by a malicious entity to make a resource unavailable to its intended users. Gligor defines denial of service as follows [Virgil D. Gligor (1949)] group of otherwise-authorized users of a specified service is said to deny service to another group of otherwise-authorized users if the former group makes the specified service unavailable to the latter group for a period of time that exceeds the intended (and advertised) waiting time." This definition of denial of service takes into account the timeliness aspect of availability, and we use it as the standard definition of denial of service. So Examples include –

1. Attempts to "flood" a network, thereby preventing legitimate network traffic
2. Attempts to disrupt connections between two machines, thereby preventing access to a service
3. Attempts to prevent a particular individual from accessing a service
4. Attempts to disrupt service to a specific system or person

1.2 Methods of attack

There are two general forms of DoS attacks: those that crash services and those that flood services. A DoS attack can be perpetrated in a number of ways. Attacks can fundamentally be classified into five families-

1. Consumption of computational resources, such as bandwidth, memory, disk space, or processor time etc.
2. Disruption of configuration information, such as routing information
3. Disruption of state information, such as unsolicited resetting of TCP sessions
4. Disruption of physical network components
5. Obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

In this paper we are mainly concern with application layer attack classified under classification named Consumption of computational resources, such as bandwidth, memory, disk space, or processor time.

1.3 Application-Level Floods

Some attacks are not designed to be located in the payload of a single message. Instead, these types of attacks operate using a series of SOAP messages. In addition to attacking an individual Web Service, these attack types possess the potential to harm the entire application server and any other WSs located on the application server. Like the Oversized Payload attack, the Flooding attack is a type of resource exhaustion or DoS attack that can attack the WS and application server. The attacker bombards the desired service endpoint with a continuous series of request messages. Each request sent to the attacked endpoint is a syntactically correct request. Since the endpoint has no way of determining that each individual request message is part of a larger attack, the WS must process each request message. Eventually, the WS consumes its resources and begins to exhibit a DoS effect on the WS.

2. LITERATURE SURVEY

The DoS attack is restrict the access to legitimate user. This attack is broadly classified in different types basically software and hardware. This attack can be prevented by using hardware like switch or router. In this paper we are mainly focused on techniques to detect DoS attack at application level and different ways to avoid it. Such DoS attack blocks the Web Service resources and doesn’t allow legitimate user to access it. These attacks are again classified in to different types i.e. message level attack and application level attack. The message level attack consists of Xml injection attack, malicious body attack, Xml retrieving attack, ping of death attack etc. The second type of attack, application level attack consists of flood attack which sends the oversized SOAP messages to the server in largest number. Our main focus of paper is on application level attack. There are different tools available in the market for the detection of application level attack like CAPTCHA, this tool is used for allowing particular request with time limit and different CAPTCHA model are made available to the user. However, this technique is not efficient because for every web page we have made different CAPTCHA and its costly and time consuming. One of the other ways is to use Puzzles system which is using different types of puzzles for every web page. So at the time encryption and decryption of SOAP message it requires different logic and hackers can’t easily able to trap. In this tool also puzzles required for every web page and there is required always different logic for every puzzle so it is time consuming tool. Intrusion and fault detection system and there different types of IDS are available. These systems can work at node of the network which can be used to detect the attack. This technique required to build on server. However, server down will bring all the application down. In the accounting model for detection of DoS attack, client cookies are used to record the request history. These cookies are used to fetch the all the required information from the client message. The model uses hash value of every incoming request. Hash value is a combination of different cookies attributes from request, which is used to identify duplicate request from client at the server. The system will fail, if a user turns off browser cookies while sending request to the Web Service. In this Paper we present target attack detection by maintaining a client's requests history and by using dynamic Web Service filters.

3. PROPOSED SYSTEM

To help in detecting and preventing the flood type DoS requests, we proposed the filter based Web Service model which will consume less computational time and memory space. The figure shows the detailed architecture of the proposed model for filtering Web Service and detecting the attack. According to this model, every client can send the SOAP request message and get the response over the Internet using the HTTP protocol. When the client request hits the server, the client request will pass through Traffic Monitoring system, then client request will be forwarded to either Request Processor to serve client request (if the Web Service hosted port network traffic is low) or to the Message attribute manager where the input request will be analyzed and attributes are copied to lightweight external database.

The Traffic Monitoring system is a module which will monitor the current number of incoming requests coming over the specified port (like command utility - netstat does on windows OS.). This will act as a dynamic filter activator. If the number of incoming requests hitting at specified service port increases beyond set point, then Traffic Monitoring system will detect it as high traffic over the network and activate Filters in chain else it assume as normal traffic and continues to serving client requests. If the request is legitimate, and the traffic is low, then request of the client will be forwarded to Client Request Processor. CRP connects to the main database for fetching data required for execution of the client request. This will generate a client response which then forwarded to the requested client over the HTTP by embedding more Information to response if required. This way Traffic Monitoring System, helps in monitoring network traffic, when there are chances of a Denial of Service attack, by dynamically activating Web Service filters. Activating Filter makes every request to pass through the Web Service filter chain where an attempt to detection of DoS attack is made.

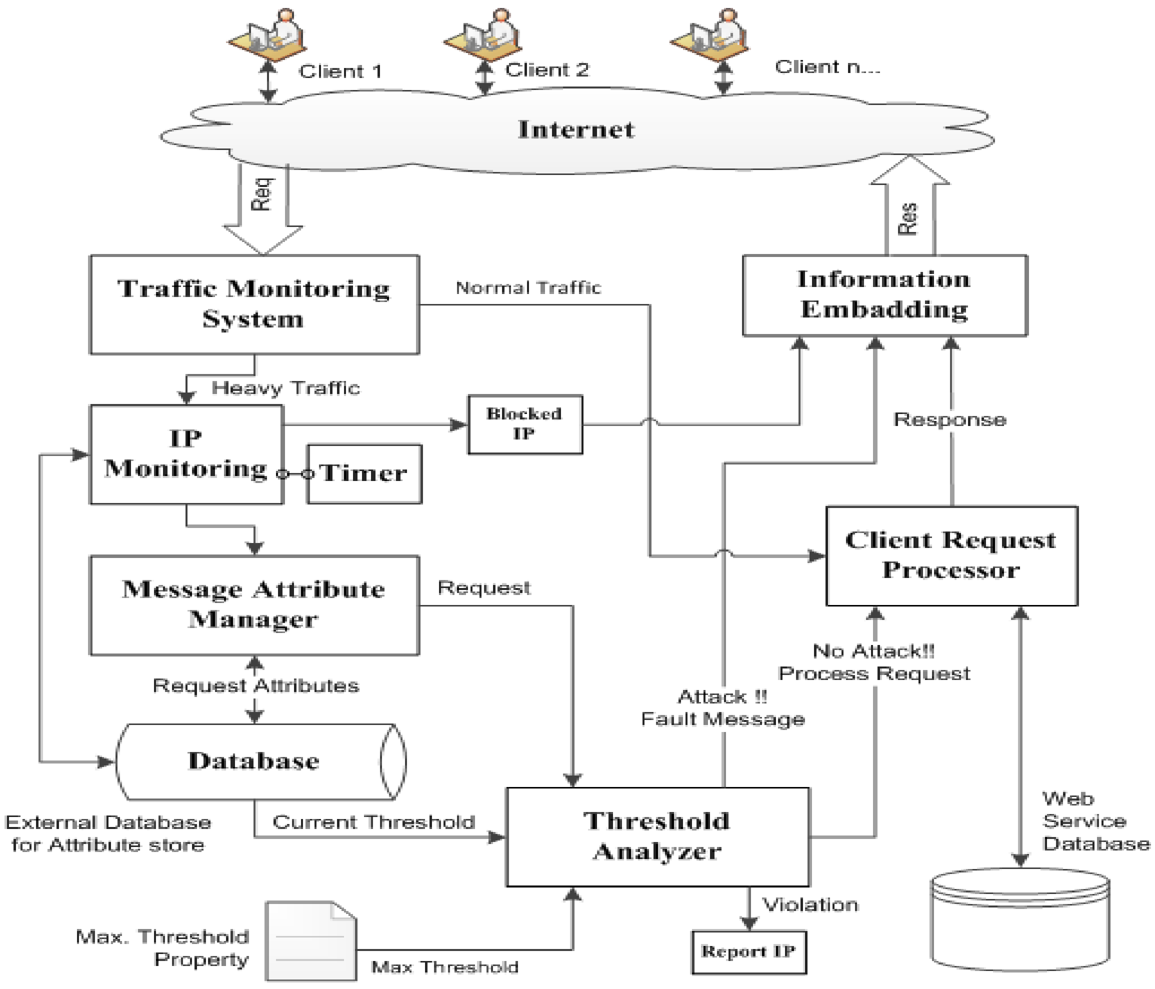


Figure 1: System Architecture for Web service DoS attack mitigation

As Flooding attack can only be detected by analyzing inbound messages, Attribute manager takes client request as input, pre-process it to examine content/attribute of input message and stores them into the lightweight speedy database system. The attribute manager is constructed within a filter chain level. Here our approach is implemented a detection mechanism that could determine how often a Web Service was receiving messages from a given client or endpoint. We designed and implemented an Attribute Manager that makes use of a database to store necessary information about all received SOAP requests. This database contains a record of the sender like sender, target and timestamp of the received message.

3.1 Request History Handling Algorithm

Step 1: Arrival of the request message and its consecutive request with SOAP message.

Step 2: Check if Traffic on the predefined node has increased beyond Maximum load threshold by using Dynamic traffic monitoring system. If yes then activate filter. Otherwise do not activate filter and forward request to Web Service.

Filter algorithm-

Step 3: Retrieve the Target from SOAP message and cache it.

Step 4: Retrieve the Sender from message header or application context and cache it.

Step 5: Retrieve the timestamp of the message and store it in cache.

Step 6: Get Current Threshold* value for respective target and user from Cache.

Step 7: Compare Current threshold value with Max Threshold* value set for application.

Step 8: If Max threshold value is greater, means no DoS attack detected, hence request will be transferred to the Web Service or next Filter in filter chain. Otherwise Filter will detect it as a DoS attack and block the request sender by adding senders IP into the Black list.

Step 9: Clear Cache objects which are no more required.

4. DATA SET

Inputs	**Details**
IP address	IP address of the client host machine.
URL	Endpoint URL
Time Stamp	The time when request arrived from client.
Request Type	Type of the input request.
Web Service parameter	SOAP Message

5. RESULTS

It is obvious that all DoS attack detection models require the additional computational time and memory than the normal scenario. The additional computational time required for information retrieval, storage, detection of DoS attack, and reading configuration files of application and so on. Similarly, this model also needs additional computational time for retrieving attributes from request message, insertion, updating and retrieval of the attributes into and from the database. The computational time complexity and the memory complexity of model are analyzed with the simple website created by us. It is developed with the HTML, JSP (Java Server Pages), JavaScript (Client side language) and Apache (Web server) with the server system configuration of 2.2 GHz processor and 1 GB RAM. We have Product.HTML page which can be accessed using two different URLs-

1. /DoSAttackMitigationWS/DoSAttackMitigation/getProductDetails/filter_on
2. /DoSAttackMitigationWS/DoSAttackMitigation/getProductDetails/filter_off

Filter ON scenario represents DoS protected Web Service. Filters are ON means Web Service is protected. Whereas, Filter OFF is a URL pattern which skip all the DOS attack protecting filters and does not attempt to detect DoS attack and give direct access to Web Service. The website consists of many other pages like –

1. Index (Home) page,
2. Product details page (Same page with two different URLs)
3. Contact us page,
4. Category-list of Product Page etc.

The user requests information is available in the java cache for this model which include individual page request count and the initial request timestamp. The light weight database record consists of the Hash Table of the request history, initial request time, host address and the Timestamps of number of requests equal to Threshold value set for Application along with list of blocked IP address. This the information available at server side for processing and detection of DoS attack.

The database table format of the DoS attack mitigation using WS filter model is shown in Table 1 and Table 2.

Table 1: Black Listed IP collection Table

Host address 1	Host address 2	Host address 3	...	...

Table 2: Light weight database client table to save client history

Host address1	Timestamp1	Timestamp2	Timestamp3	Timestamp5	...
Host address2	Timestamp1	Timestamp2	Timestamp3	Timestamp5	...
Host address3	Timestamp1	Timestamp2	Timestamp3	Timestamp5	...

The DoS attack experiment was conducted with approximate 6600 requests per minute. DoS attack is to prove the efficiency of this model. Table. 1 shows the extracted experimental information of the model with DoS attack Protection (Filter ON) and without DoS attack Protection (Filter OFF).As said earlier, Filter ON represents WS called with DoS attack protection and Filter OFF scenario means there is no mechanism adopted to filter the request to detect the DoS attack. The main intention of the experiment is to show how much processing time and memory required by these two scenarios to prove their efficiency. Also, the normal scenario is used for testing Traffic monitoring module.

Table 3: Experimental Results of the Project carried out under DoS attack.

Testing Parameter	**Web service with DoS Attack Protection.** i.e. Filter ON scenario	**Web service without DoS Attack Protection.** i.e. Filter OFF scenario
Request Count/Time	1,55,078 / 30 min	1,53,692 / 30 min
Error Count		
Overall Average Processing Time	15,71,027	684,52,534
Request Error Count with Error Code	1,54,115 HTTP 403	68,129 HTTP 500
Approximate Traffic volume	224 MB	5 GB

The experimental analysis is carried out with the memory requirements, avarage response time, Number of rejected requests of the sceanrios. Fig. 2 and Fig. 3 shows the deatils observations of the following parameters, This experiemt is carried out on both the scenarios by attacking on the Web Service with the rate of 5300 requests/min for more than half hour.

1. Number of Requests Each Interval
2. Processing Tile in each interval(in ms)
3. Traffic volume in each interval

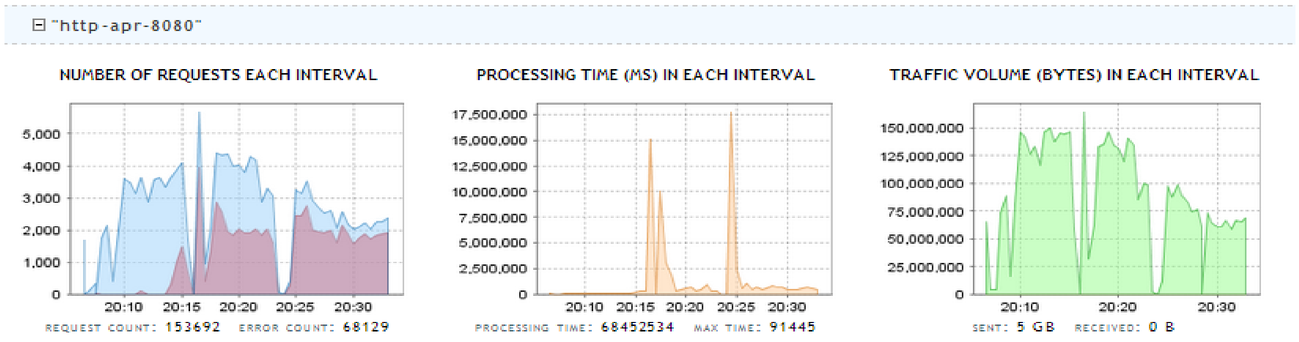


Figure 2: Application Performance without DoS attack Protection

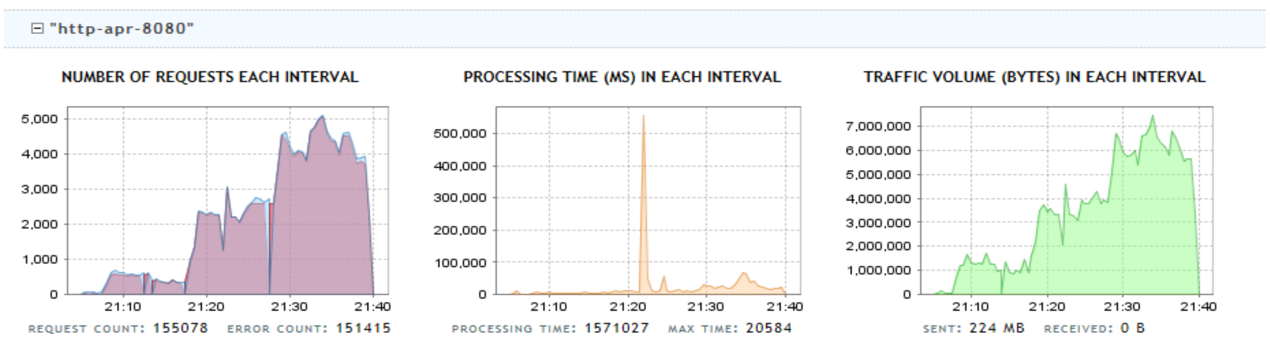
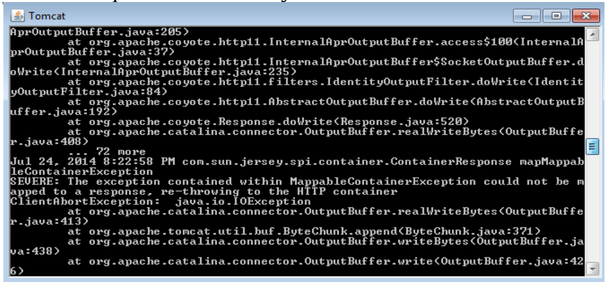


Figure 3: Application Performance with DoS attack Protection (With WS Filter ON)

Fig 2 and Fig. 3 shows that the actual results taken from the experiment attacking DoS attack on the Web Service. It shows that this model with filters consumes less processing time when compared to theFilter off scenario. Error Count is also observed high in case of filter ON scenario, this is because filters holds the capability to generate response without processing client's requests before forwarding them to the Server. Once application finds concequences of DoS attack from a particular host and then IP filter comes into the picture and starts blocking further requests from that host to the server. Resulting increasing availability of the Web application along with high performance for the ligitimate user. Again high deviation in the inconsistencyresults for Traffic Volume requested and responded from the server which was causeddue to Filter ON and Filter OFF scenarios. it shows that DoS attack mitigation Filter model takesless processing time and very less response size (in Bytes) than the model without DoS protection. Hence, the request processing time of this model is less than the model without DoS protectionhence, this model is efficient to handle the multi-millions requests sent blindly to the server to attack server.



```
AprOutputBuffer.java:205)
        at org.apache.coyote.http11.InternalAprOutputBuffer.access$100(InternalA
prOutputBuffer.java:37)
        at org.apache.coyote.http11.InternalAprOutputBuffer$SocketOutputBuffer.d
oWrite(InternalAprOutputBuffer.java:235)
        at org.apache.coyote.http11.filters.IdentityOutputFilter.doWrite(Identit
yOutputFilter.java:84)
        at org.apache.coyote.http11.AbstractOutputBuffer.doWrite(AbstractOutputB
uffer.java:192)
        at org.apache.coyote.Response.doWrite(Response.java:520)
        at org.apache.catalina.connector.OutputBuffer.realWriteBytes(OutputBuffe
r.java:408)
        ... 72 more
Jul 24, 2014 8:22:58 PM com.sun.jersey.spi.container.ContainerResponse mapMappab
leContainerException
SEVERE: The exception contained within MappableContainerException could not be m
apped to a response, re-throwing to the HTTP container
ClientAbortException:  java.io.IOException
        at org.apache.catalina.connector.OutputBuffer.realWriteBytes(OutputBuffe
r.java:413)
        at org.apache.tomcat.util.buf.ByteChunk.append(ByteChunk.java:371)
        at org.apache.catalina.connector.OutputBuffer.writeBytes(OutputBuffer.ja
va:438)
        at org.apache.catalina.connector.OutputBuffer.write(OutputBuffer.java:42
6)
```

Figure 4: Tomcat console showing error after DoS attack made for 1 hour

Fig. 4 shows the Apache Tomcat server console log when server was under attack and it went down after half hour, since attack was started and throwing exceptions to all the requests including legitimates request hitting to the server.

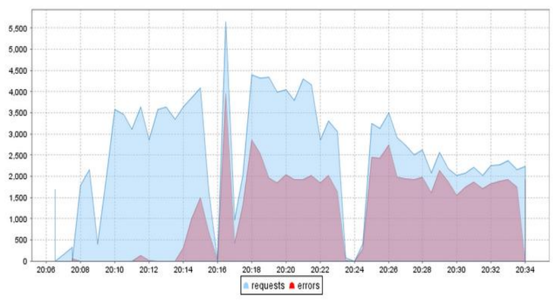


Figure 5: Application Performance without DoS attack Protection (Filters OFF)

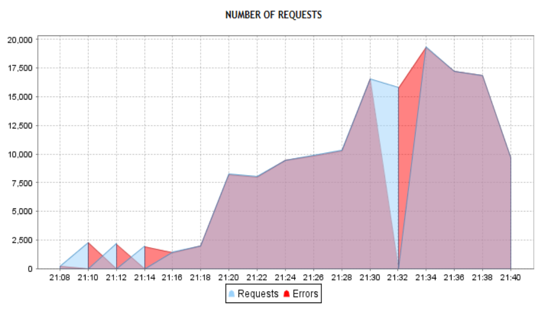


Figure 6: Application Performance with DoS attack Protection (Filters ON)

It is to be expected that some errors may occur when processing requests, especially under load (DoS attack). Most of the time it has seen that errors begin to be reported when the load has reached a point that exceeds the web application's ability to deliver what is necessary.The Error Rate is the mathematical calculation that produces a percentage of problem requests to all requests. The percentage reflects how many responses are HTTP status codes indicating an error on the server, as well as any request that never gets a response or the exception message from Server. The web server returns an HTTP Status Code in the response header. Normal codes are usually 200 (OK) or something in the 3xx range indicating a redirect on the server. A common error code is 500, which means the web server knows it has a problem with fulfilling that request and this is observed in case of Filter OFF sinario. Whereas that of course doesn't tell user what caused the problem, but fig. 5 shows that the there is a definitive technical defect in the functioning of the system somewhere. From the Fig. 6 its crear that server servers all the requets unser DoS attack and after server sturation fails to serve and response server exceptions to the user.

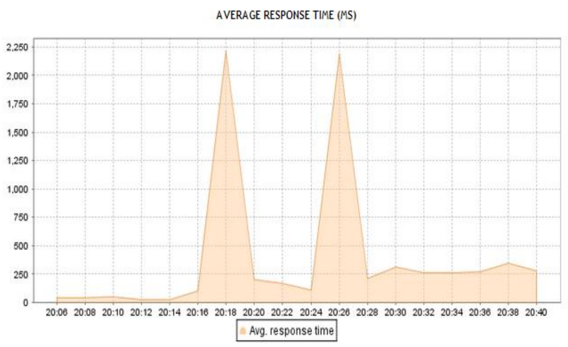


Figure 7: Application Performance without DoS attack Protection (Filters OFF)

Following enlarged graphs showing the result difference between two different test scenarios i.e. Filter ON and Filter OFF tests. The above test proved that, as too many request which found as DoS attack were rejected by the filter. The Average Response Time takes into consideration every round trip request/response cycle up until that point in time of the load test and calculates the mathematical mean of all response times. The resulting metric is a reflection of the speed of the web application being tested – the BEST indicator of how the web site is performing from the users' perspective. The Average Response Time includes the delivery of HTML, images, CSS, XML, JavaScript files, and any other resource being used. Thus, the following graphs shows average time that significantly affected by two different scenarios.

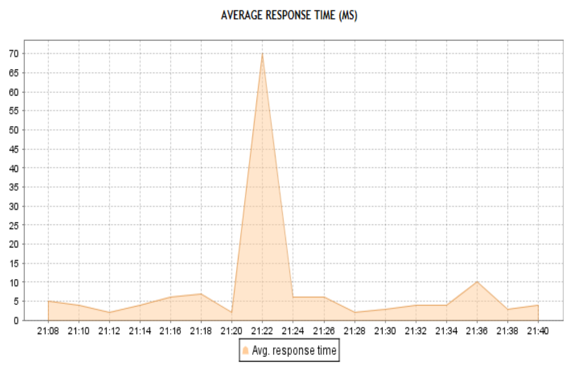


Figure 8: Application Performance with DoS attack Protection (Filters ON)

In this paper, an outline of some of the current DoS attacks that can be launched to exploit vulnerabilities within Web Services has been presented along with a comprehensive Integrated Filter based Framework for preventing these attacks. We presented a solution that uses Web Service filters to detect malicious DoS attack. Our Web Service filters combines traffic monitoring system and Threshold analyzer to mitigate and filter the potential danger of DoS attack. After analyzing existing different cookie based DoS attack accounting models, The Filter based Framework model are identified as well proven and more scalable. Later, Filter model's experimental results will be analyzed with respect to attack detection and time complexity. The future work of this paper could be to analyze traffic at network port efficiently at application level and to work effectively on addition of few more filters to cover different kinds of DoS attack.

6. CONCLUSION AND FUTURE WORK

In this paper, an outline of some of the current DoS attacks that can be launched to exploit vulnerabilities within Web Services has been presented along with a comprehensive Integrated Filter based Framework for preventing these attacks. We presented a solution that uses Web Service filters to detect malicious DoS attack. Our Web Service filters combines traffic monitoring system and Threshold analyzer to mitigate and filter the potential danger of DoS attack. After analyzing existing different cookie based DoS attack accounting models, The Filter based Framework model are identified as well proven and more scalable. Later, Filter models' experimental results will be analyzed with respect to attack detection and time complexity. The future work of this paper could be to analyze traffic at network port efficiently at application level and to work effectively on addition of few more filters to cover different kinds of DoS attack.

7. REFERENCES

[1] S. Venkatesan, M. Basha, C. Chellappan, A.Vaish, P. Dhavachelvan "Analysis of accounting models for the detection of duplicate requests in Web Services" Journal of King Saud University May 2012.

[2] XiaoFeng Wang · Michael K. Reiter "A multi-layer framework for puzzle-based denial-of-service defense" Springer-Verlag 2007.

[3] M Mehra, M Agarwal, R Pawar, D Shah "Mitigating Denial of Service attack using CAPTCHA Mechanism" International Conference and Workshop on Emerging Trends in Technology (ICWET 2011) – TCET, Mumbai, India.

[4] Yi Xie ,Shun-Zheng Yu "Monitoring the Application-Layer DoS Attacks for Popular Websites" IEEE/ACM transactions on networking, vol.17, no.1, February 2009

[5] Raja Azrina Raja Othman "Understanding the Various Types of Denial of Service Attack" .

[6] N. Gruschka, L. Iacono. Vulnerable Cloud: SOAP Message Security Validation Revisited. IEEE Int'l Conf. on Web Services, 2009.

[7] M. Jensen, N. Gruschka, R. Herkenhoner, N. Luttenberger. "SOA and Web Services: New Technologies, New Standards - New Attacks" 5th European Conference on Web Services, 2007.

[8] N. Antunes, M. Vieira." Enhancing Penetration Testing with Attack Signatures and Interface Monitoring for the Detection of Injection Vulnerabilities in Web Service" IEEE Int'l Conf. on Services Computing, 104-111, 2011.

[9] M. Ficco, M. Rak. "Intrusion Tolerant Approach for Denial of Service Attacks to Web Services". Int'l Conf. on Data Compression, Communications and Processing, 285-292, 2011.

[10] N. Sidharth, J. Liu. "Intrusion Resistant SOAP Messaging with IAPF" IEEE Asia-Pacific Conf. on Services Computing, p. 856-862, 2008.

[11] B. Yildiz, G. Fox, S. Pallickara, "An Orchestration for Distributed Web Service Handlers". Int'l Conf. on Internet and Web Applications and Services, p. 638-643, 2008.