XSD DDoS Trace Handler in Web Service Environment

A. Murugan[1]*, K. Vivekanandan[2]

[1] Department of Computer Science and Engineering, SRM Univerisity, India.
[2] Department of Computer Science and Engineering, Pondicherry Engineering College, India.

* Corresponding author. Email: murugan.abap@gmail.com
Manuscript submitted April 4, 2015; accepted July 28, 2015.
doi: 10.17706/jsw.10.9.1086-1095

Abstract: Web services became a crucial tool for most of the Internet and Intranet applications and distributed systems due to its interoperability. As the usage is increased, performance and the security of Web services are facing a great impact due to DDoS attacks, XML Injection, XSS Injection etc. The hacker's major target is either to track the data down the line or to break the network bandwidth and feed in vulnerable data to collapse the system. Existing trends follow a stream based approach with encryption techniques to increase robustness, or a Double Guard Intrusion Detection System with light weight virtualization is adapted to prevent the attacks over multitier web services. Various prevalent techniques focus towards securing data, increasing robustness and improve the network bandwidth. But there is no focus towards the validation of the service request. The proposed XSD DDOS Trace Handler approach is an innovative framework that uses the concepts of Validate Handler for the input request based on input data standards and request timestamps from the specific host. It also implements the "Totient Encryption Algorithm" in the case of XML Injection Attacks wherein a clean monitoring of the source attributes is performed. This is achieved with a dynamic charting technique that overcomes the predominant injection and DDOS attacks in the service oriented architecture.

Key words: Web services, XML security, attacks, XML injection, XSS injection, schema validation, schema hardening, WS-security, SOAP messages.

1. Introduction

Web services are increasingly used by the distributed systems on the Internet. They provide a standard means of interoperation among different software applications running on a variety of platforms and frameworks. Web services [1], [2] are extensively used in significant part, of the Web as they are simple to use, platform independent and supported by various XML Standards thus leading the applications to communicate easily according to Fig. 1. However, the underlying technologies used by Web services [1], [2] such as SOAP, HTTP, and XML have fostered the deployment of well-known vulnerabilities in the new environment.

There is a wealth of research on web service attack types and different techniques to mitigate them. Quite a lot of techniques have been implemented in application servers to detect web service attacks. Intrusion Detection System has been used in various kinds of researches for detecting and mitigating attacks. It detects flooding, coercive parsing; oversized payload and various types of injection attacks, according to Fig. 2. The use of web service security, schema validation, web service security policy enforcement and schema hardening are some of the vital prevention mechanisms. However, there is little discussion on reusable

methods for implementing these known techniques. Also, there is a lack of client request validation methodology.

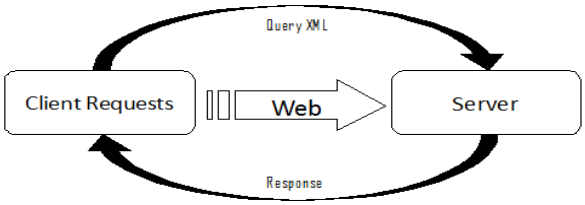


Fig. 1. Request vs response in SOA environment.

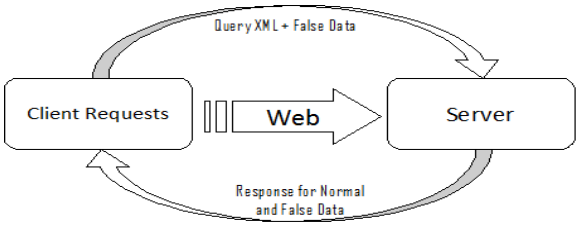


Fig. 2. Request vs response in SOA – attacked environment.

2. Literature Survey

Business processes are more dependable on Internet services. These services are subject to a number of vulnerabilities. Service Oriented Architecture provides a flexible composition of local and remote services with very high degree of integrity composition factors which includes availability and confidentiality for the end users. Vulnerabilities and security attacks become more predominant and incur high costs on the victim's side. The vulnerable service will not only consume the cost of the service and it should incur the penalties and compliance violation cost too. Lutz Lowis and Rafael Accorsi [3] provides an innovative AT List vulnerability report analysis, which segregates the exploits and cleared out the detection technique by subdividing the entire attack into multiple elements such as Point of View, Attack Effect, Active Component, Involved Standard and Triggering Property. The Communication between User Interface (UI) Server and Database (DB) Server becomes the common architecture in the wholesome internet applications. Mixing Le, AngelosStavrou [4] analyzed the omnipresent nature of internet based client server application and a Double Guard Intrusion Detection System with light weight virtualization is adapted to prevent the attacks over multi tiered web services [1], [2]. Using this technique, an effective two level scrutinization which includes a front end validation and backend validation and an intermediate ID between the two tiers will take care of communicating in the proper direction to safeguard the system. Creating online overflow attack blocker is one of the prominent ideas to overcome code injections in web services [1], [2], [5]. Sigfree technique provides a new O (N) algorithm which can disassemble the executable binary code and distill all the possible instructions to identify the malware based executable codes. In case of missed out scenarios in the above technique, the Code Abstraction technique provides an effective solution by comparing the Useless Vs Useful instructions in the executable code. Nils Gruschka, MeikoJensen [6] introduced a stream based security approach for the web service to handle various encryption/signatures. Once the xml document is passed through the web service, a stream based approach is defined to validate the xml through XML Schema validation. In case of violation, the server will immediately cut off from the client to avoid any other effects due to the predicted exploit on the system. But this approach failed to achieve high robustness and hence requires implementation of configurable processing chains with components for handling attachments and for checking message sequences for BPEL-composed web services [1] [2]. Various other techniques like XML Similarity classifier [7] provides much more robust solution in case of XML based attacks. Code level XPATH injection becomes predominant in the web service attack world and this can be restricted by validating the XPATH values [8] AziahAsmawi, Lilly SurianiAffendey, NurIzuraUdzir, RamlanMahmod didn't provide a clear picture on the algorithmic part in identifying the XPATH attack. Apart from retrieving secure information, many attacks were reported in the field of Denial of Service arena. DDOS involves restricting the resources to intended users by flooding fake requests on the system. XiaoFeng Wang and Michael K. Reiter analyzed the fact of DDOS and introduced WRAPS concept by providing the

referral links to the legitimate users. So that, the users are not intended to access the websites/secure links directly. In [9] the technical front, routers will restrict and enable more protection by dropping the packets from the untrustable resources. The drawback behind this scenario is, the websites should have a very big referral network and it won't suits mid level company websites. In part to this idea, Yajuan Tang, XiapuLuo proposed a feedback control based implementation which will send high intensity requests in an On/Off pattern to reduce the impact of the DDOS request coming into the system. The overall idea proposed to reduce the impact of the attack and not focused effectively on restricting or stopping the attack. This provides a clear pathway for our research towards restricting the attacks by monitoring the Request vs IP Address vs MAC Address.

3. Traditional Approach

3.1. Inference Attacks

Web Service is used to build complex Business Service Oriented architecture based systems. These service based architecture sectors relies on exchange of XML through various factors such as loosely coupled architecture and interoperability (Tendency of the web service to be compatible with a variety of technologies in providing the output while accessing).

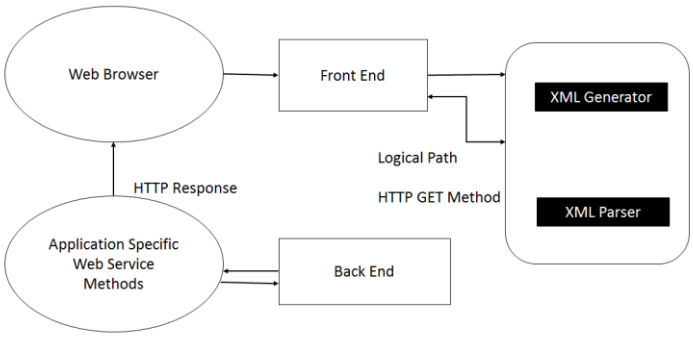


Fig. 3. Web service specification in application tier.

In Fig. 3 the web service engine holds a XML parser to fetch the required data. Exploiting this parser by amending new nodes and tweaking the input content for the XML provides a fantastic path to inject the data into the system or to extract the data from the system. XML Signature provides an optimal solution towards web service attacks. This enables low software detection mistake rates, but the major drawback with this kind of detection technique is, it won't detect new, unknown detects in case of small variation in the known payload. To overcome this behavior, the knowledge based detection system is introduced which can redirect the request vs response type into specific categories. The first class is assigned for normal behavior in which the action is already recorded and there doesn't issue with the execution, whereas the second class is related to the unexpected behavior to which the admin defined action will be taken or there won't be any response for this kind of request. Still, the issue is with high false-positive rate.

Existing methodology combines both the above said techniques by a signature based system aligned or interlinked with catalog database which can repositories all the possible attacks happened on the system. The previous attacks and attack elements were represented by classes and relationships and maintained in an Ontological database. The XID based system is mitigated which can store all the relevant captured events. Considering an Attack Action instance represents an attack, the knowledge based system will search for the possible attacks list and check the matched context. If the attack is not matched, the reasoner will take care of inferring the set of captured events as an instance of a specific class (For ex: XQuery Injection compiling with the defined axiom) and add the attack details and scenarios into the Ontological database after

learning the new attack instance behavior.

The XID will take care of storing all the new attack scenarios as sub categories. Very less re-modification of the classes and relationships is one of the major advantages of the system. The reason is, each individual attacks were modeled independently and has its own profile. In case, if the strategy defines or resembles an odd behavior, the inferred new attack instance, will be certainly grouped under the attack and its repositzed in the system for future references.

3.2. Detection Mechanism

XML Data becomes too famous in a low weight protocol based data sharing between the client and server systems. Since, the performance of the process has become the major key role in the SOA architecture, world - most of the business end to end applications works with an underlying XML based data transfer. The XPath expression in Fig. 4 is one of the key techniques used in accessing XML Data. Most of the predominant data vendors are now started supporting XML based Data and in addition, the XPath expression is to access the same. XPATH injection attack is considered as one of the most vulnerable attack due to its over exposure of entire data at a shot without restricting to specific data during XML Data exploit.

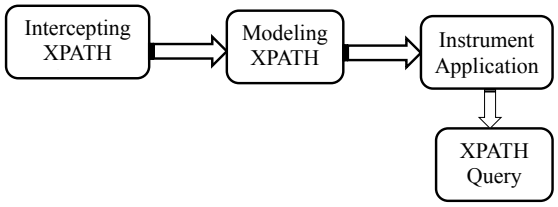


Fig. 4. XPath expression.

The overall detection process involves in identifying the hot spots in the application code. After identifying the hotspot, a model is built to identify the possible XPath Query combinations that are generated in the hotspot and validate the same by providing a monitoring code to check out before it hits the database for querying. During validation, if any, of the query didn't abide by the standards, it will be rejected.

Providing referral links to the users to access the server is one of the predominant options proposed in the Denial of Service attacks. Monitoring the application access from specific IP Addresses and the specific user account is one of the technique Followed by, reducing the impact of resource usage by the specific request zone is one of the nominal path on Denial of Service attacks.

4. Materials and Methods

4.1. Xml Injection Attack — Detection/Prevention

XML injection attacks—those that turn out some change in the XML's internal components, aims to compromise the Web service application. The data in a Web Application is stored in a XML database comprised of XSD and it has been accessed by the XPath Generation method. In this method, an XPath query is generated after the user provides the input to the system and the required data is accessed. The problem arises when the input provided by the user is not properly filtered by the system. This kind of XML Injection can be achieved by injecting malicious content into an XML message, such as invalid XML characters. When the user request the web service in order to claim the web service response action, the XML Elements are being encrypted Using "TotientCiphering Algorithm" which will encrypt the user's data in the form of XML Schema by preventing the data's from attacking criteria. If an XML document is not validated as "Well

Formed" or checked against a schema, risks may arise which will allow the attackers to attack the web service. So, in order to prevent the XML injection attacks following measures will help out possibly with the ScrutinizationSteps as follos.

1) Utilizing the User Data from the web service access point
2) Generating XML for the User data that possess the XSD
3) Checking For Valid XML Schema Definition
4) Validate/Sanitize User data by strong Totient Ciphering Encryption to avoid XPATH injection
5) Service Provision to Appropriate User and denying the Unsolicited User.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelopexmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
</soap:Body>
</soap:Envelope>
```

4.2. DDOS Attack — Detection/Prevention

The Virus is a small application that consent to remote commands and control, competence of the computer without the acquaintance of the users. Denial of Service (DOS) compromises a single system that is infected with a type of virus application known familiarly as Trojan. Distributed Denial of Service (DDOS) is a kind of DOS attack that compromises multiple systems that are infected by Trojan virus. DDOS craft's brutal destruction of the system by the commands from the remote source acquired from the multiple systems to a single system at an instance. The communication chamber of a client-server model ruined with the DOS attacks that are hosted on high profile web servers.

L7 (Application Layer 7) of DDOS generates a minimal level of network traffic, but their destruction the website heavily. They may activate web application by changing the username and passwords. The biggest issue is that Layer 7 attacks change and randomize rapidly. In Fig. 5 the Victims of a DDoS attack consist of both the end targeted system and all other systems that have been maliciously used and controlled by the hacker in the distributed attacking state of affairs.

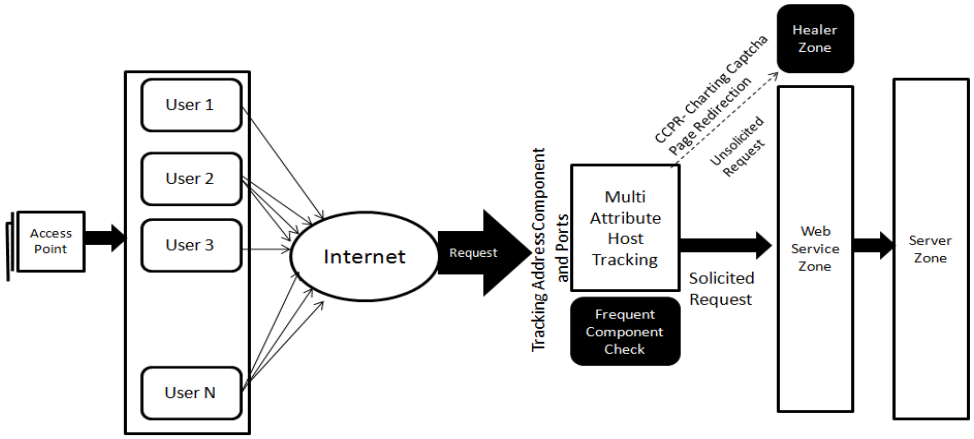


Fig. 5. DDOS trace handler.

The proposed system dealt with the DDOS attack of tracking the unsolicited user's traffic by employing

traffic policies by authorized admin. The framed policies of admin fix a threshold to hold back the attack traffic using baseline profile. The baseline profile comprises of predetermined rules to monitor the traffic conditions. The monitored traffic over the bounds avoids the delay of service provision in terms of response by traffic shaping methodologies framed in the baseline profile [10]. In the traditional mechanism of the DDOS detection, the raised requested is analyzed by the IP address of the requestor user. Since the IP address can be changed or by using virtual machines, the detection of DDOS is not effective. The detection of the unsolicited user is identified by the request raised from the IP address, Port Number of the request as well as the main key component of user identification, the physical address of the system. In the intervention of the user in the access point to raise a request to the web service zone is monitored by the user session with the address components to restrict the unsolicited user with their identity.

In a client-server model, an incoming user in the structure of XML-Schema via web services [1], [2], [11] is difficult to distinguish between the legitimate and UnSolicited users traffic using point of access from the origin. An effective algorithmic mechanism called Multi Attribute - Host Tracking has been utilized to track the communication components of the users such as IP Address, MAC Address and Port Address and act as a detection mechanism. Healing with the detection mechanism, a preventive technique called CCPR Algorithm (Charting Captcha Page Redirection Algorithm) has been incorporated by Page redirection Algorithm to identify the vulnerable injections occurred; prevent the adequate request to the web service. The recent advent of request prioritization is furthermore considered as a DDOS attack, wherein the unsolicited users place an incorrect operation in the SOAP body thereby using a different SOAP header. If a condition occurs to overrule the web services [1], [2] that prioritizes the SOAP body over SOAP Action header, the unsolicited users places the incorrect operation in SOAP body using different SOAP header. The Modification apprehends in the header and the body of the SOAP dodges out the HTTP filtering system.

Here below, the operation WSDL has been explained that perform the “Stock Details” in order to act upon the SOAP request.

Example:

```
<wsdl:operation name="FetchPortComp">
<soap12:operation soapAction="http://tempuri.org/FetchPortComp" style="document"/><wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="FetchStockDetails">
<soap12:operation soapAction="http://tempuri.org/FetchStockDetails" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="FetchTotalSaleStock">
<soap12:operation soapAction="http://tempuri.org/FetchTotalSaleStock" style="document"/>
<wsdl:input>
```

<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>

5. Algorithm

5.1. Totient Ciphering Algorithm

Let I be the original message to be transferred, K is the master key component that generates exponent's e and d to cipher the original message and encipher the cipher message.

Ciphering the content can be achieved through the Totient Ciphering Technique.

$Cipher_{comp}$ ($Cipher_c$): I mod (KeyComp-1)

Deciphering the content from the ciphered text can be achieved through.

($Decipher_c$): $Cipher_c$mod (KeyComp-1)

Chose a distinctive prime number Prime1 and Prime2 to compute Master key component KeyComp. Set the Decrementive product of the chosen distinctive prime number as KeyComp'

$KeyComp = Prime_1 * Prime_2$

$KeyComp' = (Prime_1 - 1) * (Prime_2 - 1)$

Compute **$Cipher_{comp}$**, the ciphering component such that the **$Cipher_{comp}$** should be lesser than **KeyComp'** and **$Cipher_{comp}$** should be greater than unity.

Compute **$Decipher_{comp}$**, the deciphering component satisfying the constraints of **($Decipher_{comp}$ * $Cipher_{comp}$) % KeyComp** is unity

Selecting the **$Cipher_{comp}$** and **$Decipher_{comp}$** should satisfies the condition of **$Cipher_{comp}$<$Decipher_{comp}$** .

The Resultant value **$Decipher_{comp}$** is equivalent to **I** and the postulate is verified for its correctness.

5.2. Experimental Prework

Consider $Prime_1$= 13, $Prime_2$= 11

$Cipher_{comp}$= 7, $Decipher_{comp}$=103.

Taken **$ICipher_{comp}$**= 5,**($Cipher_c$):** 25, **($Decipher_c$):** 5, The Size of the key is considered to be 2048bits as the lower 1024 is least effective and higher 4096 is less secure. Memory utilization is better in terms of storage allocation with 2048 bits.

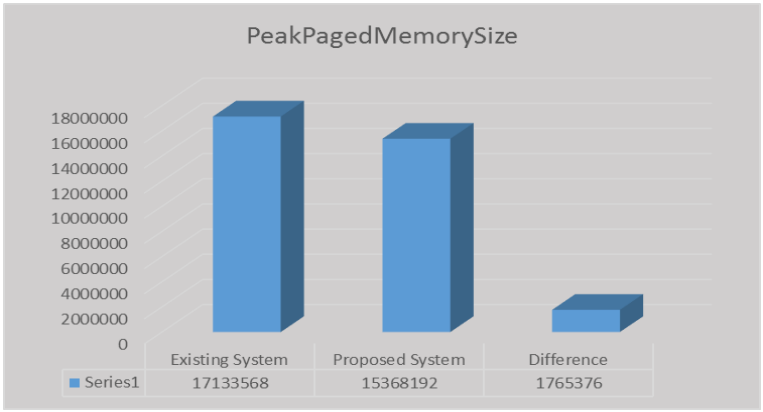


Fig. 6. Comparison representation of peak paged memory size utilization of algorithm.

5.3. Charting Captcha Page Redirection Algorithm

Input: Physical Address of the Machine.

Step 1: Create a Bitmap image with Admin defined Width and Height(For ex: 100 pixels Image).

Step 2: A graphics object needs to be instantiated for the Bitmap and smoothing mode for the graphics object needs to be initialized.

Step 3: Generate a Rectangle with required Width, Height and Co-ordinate Position by utilizing the graphics object.

Step 4: Utilize HatchStyleString and generate HatchBrush object with color component to fill the graphics object.

Step 5: Set up the text font with font family, color, size and set up the String format and Alignment for the String.

Step 6: Create a GraphicPath object path and Add the String to the GraphicPath object with Font Specifications.

Step 7:Generate random Noise and Add random noise to the Bitmap image.

Step 8: Dispose HatchBrush, graphics, object and Font Specification component.

Step 9: Return bitmap Image.

6. Results and Discussion

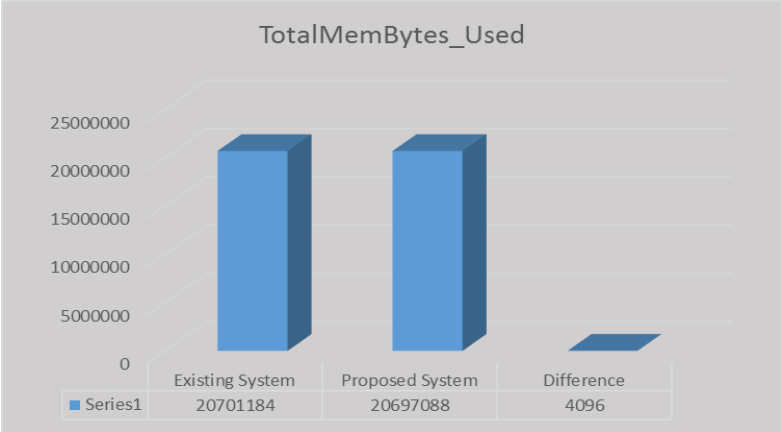


Fig. 7. Comparison of memory utilization of algorithm.

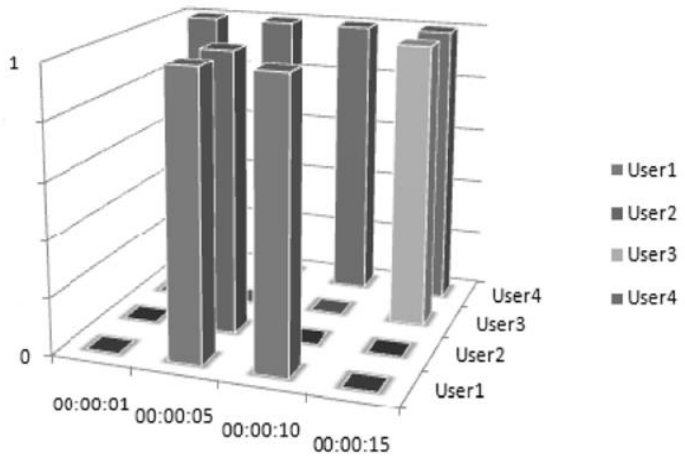
In the Graphical Representation of Fig.6 The Existing Algorithm utilized a 17133568 Bytes of Peak Pages Memory and the Proposed Algorithm utilized a 15368192 Bytes of Peak Page Memory.In the Graphical Representation of Fig. 7 The Existing Algorithm utilized 20701184Bytes of total Memory and the proposed Algorithm utilized 20697088 Bytes of total memory. The Difference of 4096 Bytes has been retrieved. This representation showcases the efficiency of the proposed algorithm in terms of Memory Utilization.

6.1. DDOS Trace Handler Analysis Report

In the Fig. 7. A Three Dimensional Graphical represents the DDOS Identification. The User request is taken as 1 and non request is taken as 0 with Seconds from 1 to 15 for 4 users namely User 1 to User 4 as Sample data. The acquaintance of many user requests for the successive 15 Sec is considered to be the unsolicited DDOS attackers, User 4 as per the representation.

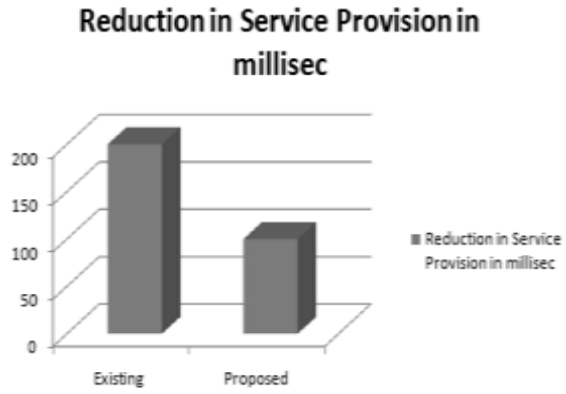
The remaining users are considered to be the solicited users. In the Fig. 8. (a) and Fig. 8 (b) The Graphical Representation Service Restriction is charted. A Maximum level of Service Requisition is considered to be 1 Request for 5 seconds, hence 12 requests for 60 seconds. The less number of restrictions thrives the server to be more secure. According to the existing postulates the Minimum 5 request per 60Seconds is considered to be DDOS attacker. The proposed postulates the minimum of 3request per 60seconds to be DDOS attacker

for redirecting to the Healing Zone. This process is carried out for every 300 Seconds to redirect the unsolicited by making Frequent Component Check. In Fig. 8(c) The Graphical Representation of Efficiency with respect to millisecond is done. The Existing Postulates utilizes a delay of 200milli seconds for every 3600 Seconds of Service Provision in responding other solicited request during the DDOS Identification. But the Proposed Postulates utilize only a delay of 100milli seconds for every 3600 Seconds of Service provision in responding other solicited requests.



(a) DDOS identification

(b) User request



(c) DDOS attack

Fig. 8.(a) DDOS Identification (b) Service Restriction Comparison of Existing and Proposed with Respect to User Request and Min (c) Efficiency Comparison of Existing and Proposed with respect to Service Provision in Millisecond during DDOS Attack.

A statistical report of the overall cross-mitigation occurrence to the fields by DDOS is surveyed from various report sources collectively from real-time scenarios. Amongst 87% of diverse field sectors were attacked multiple times.73% of DDoS attacks were under 1 Gbps relates the traffic over bound ,55% DDos attack were considered to be the data breach on the intellectual property, More than 40% estimate DDoS losses in wild sectors acutely $1MB+ per day.25% of overall attack utilized 1-5 GB of bandwidth.

7. Conclusion

As a discerning wide-ranging postulates can obtain a high-quality resultant tracer in the area of web service security measures. The problem with the existing system for identifying an ineffective request provision through IP address and XML injection detection with low weight Protocol can overcome by

proposing an effective methodology of utilizing the physical address to detect the unsolicited user and examining the XML with schema and by encoding strategies. Our Study and postulates with improved set of formulae found to be optimal option than the existing scenarios.

References

[1] Raja, B. K., & Kaliyamurthie, K. P. (2014). Trustworthy management for safety banking using light weight BFT method. *International Journal of Inventions in Computer Science and Engineering*, *1(2)*.

[2] Khushboo, R. S., & Saravanan, T. (2014). Radio frequency technology for intelligent transportation system. *International Journal of Inventions in Computer Science and Engineering*, *1(2)*.

[3] Lutz, L., & Rafael, A. (2011). IEEE vulnerability analysis in SOA-based business processes. *IEEE Transactions on Services Computing*, *4(3)*.

[4] Le, M., Angelos, S., & Brent, Byung, H. K. (2012). Double guard: Detecting intrusionsin multitier web applications. *IEEE Transactions on Dependable And Secure Computing*, *9(4)*.

[5] Wang, X. R., Pan, C. C., Liu, P., & Zhu, S. (2010). Sig free: A signature-freebuffer overflow attack blocker. *IEEE Transactions on Dependable And Secure Computing*, *7(1)*.

[6] Nils, G., Meiko, J., Luigi, L. I., & Norbert, L. (2011), Server-side streaming processing of WS-security. *IEEE Transactions on Services Computing*, *4(4)*.

[7] Mahdi, B., Belinda, F., & Ali, S. (2011). Web service, intrusion detection using XML similarity classification and WSDL description. *International Journal of U-and e-Service, Science and Technology, 4(3)*.

[8] Aziah, A., Lilly, S. A., NurIzura, U., & Ramlan, M. Model-based System Architecture for preventing xpath injection in database-centric web services environment.

[9] Sasikala, S., & Bhuvaneswari, R. E. (2014). Virtualized cloud computing using hypegear disk I/O model. *International Journal of Inventions in Computer Science and Engineering*, *1(4)*.

[10] Tony, T. N. M., Albert, K. T. H., Lee, W. L., Daniel, X. P. L., Alan, K. L. C., & Judy, W. S. W. Universal DDOS mitigation bypass.

[11] Pratheeba, R., & Purushothama, R. (2014). Modeling smarty web search engine using Xml clustering. *International Journal of Inventions in Computer Science and Engineering*, *1(2)*.

A. Murugan completed his M.Tech in computer science and engineering from Bharath University in 2005. Currently he is working as a assistant professor at SRM University, Chennai. Currently he is pursing research in SOA and web service security.

K. Vivekanandan obtained his B.E degree in electronics and communication engineering from Coimbatore Institute of Technology in 1986, the M.Tech in computer science and engineering from Indian Institute of Technology in 1991 and the Ph.D in 2005 from Pondicherry University in the area of software engineering. His teaching, research and development interests have been in the areas of software engineering, object oriented systems, DBMS and data warehousing, Information security and data mining.