Reducing the Effect of Denial of Service in Web Service Environment

Abdulrahman Alshayea [1,*] and Mohammad Ali H. Eljinini [2]

[1] Department of Software Engineering, Isra University, Amman, Jordan
[2] Department of Computer Information Systems, Isra University, Amman, Jordan;
Email: ma.eljinini@iu.edu.jo (M.A.E.)
*Correspondence: abo_od_x@yahoo.com (A.A.)

***Abstract*—Denial of Service (DoS) attacks can cost online and web service providers money and damage their reputations. The lack of security protection in web services creates a vulnerability attackers can exploit. A new XDoS attack targeting web services has recently emerged, using XML rather than plain old HTML as the attack vector. This paper proposes a middleware tool for detecting and preventing web service XDoS and HTTP flooding attacks. A rule-based technique classifies requests as benign or malicious to detect XDoS attacks. According to the middleware tool's trial findings, rule-based technology has successfully recognized and blocked XDoS and HTTP flooding assaults such as large payloads, forceful parsing, and external XML elements in near-real time, such as 0.006s across web services. Middleware protects web services from XDoS and distributed XDoS attacks by ensuring nearly 100% service availability for routine requests (DXDoS).**

***Keywords*—denial of service, web service, attack**

I. Introduction

Web Services (WS) are implemented in various scenarios, including small and large-scale businesses. Self-descriptive components that can be used by other software across the Web can be provided by Web Service Description Language (WSDL) and the Simple Object Access Protocol (SOAP)-based web services that support standard protocols like SOAP and WSDL. Web services based on WSDL and SOAP are now essential components of system integration. Web services provide a simple, well-defined interface for providers and consumers, consisting of actions and input/output parameters specified by the provider [1–3]. Various technologies and management processes ensure the confidentiality, integrity, and availability of information transmitted over the Internet. This paper discusses Web services security and the testing and validation techniques needed from development to deployment [4]. Many general security risks may exist in Web service security issues; however, evaluating general threats in any Web application is essential before analyzing security concerns specific to a Web service [5, 6].

Manuscript received October 5, 2022; revised November 21, 2022; accepted February 1, 2023; published October 20, 2023.

In this paper, we first investigate and evaluate the security of web service frameworks. Then, we propose an anti-XDoS and anti-HTTP flooding solution that uses a middleware tool to detect and prevent real-time XDoS and HTTP flooding attacks. Finally, we assess the proposed method using the two scenarios we developed. The contribution of work is as follows: the accessibility of online services is a crucial aspect of company continuity. Therefore, XDoS attacks are a legitimate concern for online businesses. When an XDoS attack occurs, the system becomes inaccessible to its intended users because all available computing resources, such as the CPU and RAM, are utilized. Experiments with XDoS attacks in this paper demonstrate that even with a modest allocation of resources, they can bring down a web server. Testing has shown that the tool effectively detects and prevents such attacks. In addition, the middleware tool can detect and prevent XDoS and flooding attacks in real-time. In addition, the tool's availability for routine requests is nearly perfect.

The structure of the paper is presented as follows: Section II presents related work on Denial-of-Service (DoS) attacks caused by a communication outage. In Section III, we go over our research methodology in detail. Section IV contains a discussion of our work. Finally, in Section V, we present our work's evaluation, and in Section VI, we provide a conclusion and future work.

II. Literature Review

Many technology experts regard web services and service-oriented architectures as two of the most significant developments of the last decade. Most resulting attacks are of the "Denial-of-Service" (DoS) variety. The magnitude of Distributed Denial of Service (DDoS) attacks on the Estonian government and commercial websites in April and May of 2007 exemplifies this perfectly [7]. The attack used computer network flooding tactics, specifically botnets. Attacks on web-based denial of service require fewer resources than attacks on most networks. It is possible to hit a wide range of targets. Services, Web services with WS-Security, and Web-oriented Web services. Web Service variants (such as WS-BAG) apply to all Web Service compositions, but for illustration purposes, we have chosen WS-BP (or BPEL,

doi: 10.12720/jait.14.5.1096-1102

for short) as an attack paradigm. Everything, however, is subject to change [8, 9].

Cloud Computing's new approach poses a new level of protection problems. First, Grobauer [10] revealed security issues related to cloud computing. Second, the weaknesses are: (1) VM escape; (2) session hijacking; (3) unsafe cryptography; (4) interface connection to the management system; and (5) online protocol security flaws, including online data retrieval and extraction. Third, the review concluded that existing protection measures are not tailored to Cloud infrastructures, so new measures are necessary. While they discussed Cloud-specific security threats, no viable alternatives are offered. In Ref. [11], the researchers identified, organized, interpreted, and measured cloud data-taxology: network design, device protection, data processing problems, and storage rights. They often used pie charts to illustrate protection threats and countermeasures. Security issues are considered 12%, and less than a third of the work is being done on possible alternatives. It means finding new ways to separate VMs from each other to prevent hardware attacks (CPU, storage, memory). External Cloud firewalls shield the provider's networks against insiders and external threats, mitigating Denial-of-Service (DoS) and Distributed-Denial-of-Service (DDoS) assaults. We have found three types of Cloud Security: as a Service (SaaS, PaaS, and IaaS) [9]. They used a test to investigate Cloud Computing risks and an assessment implementing machine learning methods to uncover them. In Ref. [12], the authors identified, classified, and listed the various security problems, weaknesses, strategies, data security procedures, and platforms needed for the SaaS, PaaS, IaaS, and OaaS models. The paper detailed many types of attacks: service hijacking, code theft, DDoS, and memory problems on virtual machines.

Khalil *et al.* [13] categorized cloud threats into standards, control, infrastructure, data, network, and networking standards, as well as defense (IMS). Only intrusions that have been detected are studied (IMS). Ali *et al.* [14] showed that connectivity and cloud protection problems can be presented at the level of the virtual system, the client-server level, and the user level. Various countermeasures have been discussed to deal with the protection problems. They used tables to highlight the various countermeasures and provided additional information on each one of them. In recent research on the forms of Distributed Denial of Service (DoS) attacks in the cloud computing world features several virtual machines and hypervisor threats [15]. Denial of Service and cloud threats are also used in the authors' well-known network and infrastructure security strategies. In an aggressive denial-of-of-service assault against a cloud infrastructure survey [16], an application-level assault is split into two categories: infrastructure and creative. Additional cloud computing capabilities (centralized control, resource access, flexibility) are required to protect end-users and cloud resources. Protection depends on the consumer or the supplier: depending on the model. It could be one or the other. As mentioned above, cloud computing protection is becoming well-understood. Our investigation explicitly covers Distributed Denial of Service (DDoS) and DDoS assaults on cloud computing. The following pages covered cloud infrastructure attack modes and security models.

Abdelsayed and Glimsholt *et al.* [17] have suggested a system to analyze network traffic using heuristic principles such as tabulated packet statistics. This procedure uses several pre-configured tables to identify domain and IP-address imbalance. In Ref. [18], it has also been suggested to implement a way of detecting DDoS attacks in local networks by measuring the flow entropy of the network routers and reporting if it is missing. In addition, Flash mobs are distinguished from Distributed Denial of Service (DDoS) attacks because of the lack of knowledge distance. In contrast to previous studies, the current approach relies on data and hypotheses. At the beginning of this process, the analysis of the local network's entropy is performed, and if the value drops over time, an attack signal is generated. Suspected network flows result in a decrease in router flow entropy. In Ref. [19], a new Distributed Denial of Service (DDoS) mitigation method is built on teamwork, and attacks may be deterred or addressed at the root, considering the attacker's location and the attacked locations. Anitha and Malliga [20] employed CLASSIE, a rule-based detection framework, to identify XDoS assaults. The packet tagging approach was used to avoid spoofing attacks. There should be a mechanism one hop away from the host to recognize known floods and XDoS assaults. In addition, CLASSIE drops packets that match one of the rule sets. Edge and core routers flag packets that have passed CLASSIE testing. One bit is needed to show that a packet has been tagged at the edge router.

III. Materials and Methods

The web service engine requires an XML parser to parse an incoming SOAP message. An attacker could use this parser to send malicious SOAP requests for XDoS attacks. Including a Document Type Definition (DTD) in an XML document may cause the parser to behave differently [21, 22].

According to a recent study, XDoS and DXDoS are more harmful than traditional DoS [23]. SOAP requests containing malicious XML content are sent as part of these attacks designed to consume system resources. As a result, because these malicious requests appear to be legitimate packets, TCP/network IPs or transportation layers fail to detect them. Because firewalls cannot scan XML content for suspicious packets, stopping malicious traffic with firewalls is difficult.

As a result, this research aims to fill that void by presenting a middleware solution that uses deep XML analysis techniques to identify and mitigate flooding attacks at the application and network levels of the OSI model. This method is based on the notion that XML packets can be analyzed in two ways: content-wise and structure-wise. Furthermore, using a rule-based classification approach, the middleware tool classifies these packets as benign or harmful.

With the proposed technology, intelligent detection and prevention of XDoS and HTTP flooding attacks in web

services are possible. The suggested tool simplifies implementing the strategy on any public or commercial web service-based application. The utility can be deployed using web service and middleware servers. The proposed tool has two main parts the rule-based classification module and the mitigation module shown in Fig. 1.

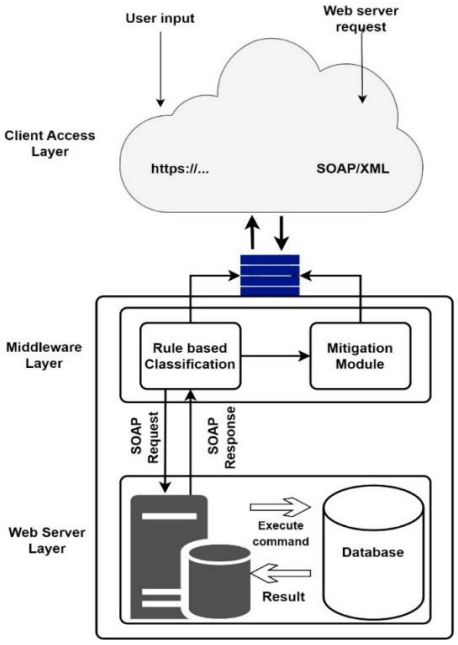

Figure 1. System of proposed work.

Fig. 1 depicts three primary levels: the client access layer, the middleware layer, and the web service layer. First, one or more web services may be clients to the web service providing the client access layer. Second, the middleware layer receives the HTTP/SOAP request from the client layer and categorizes it according to predefined criteria. Finally, the web service layer receives and processes soap requests before returning a SOAP response with the necessary results to the original requester.

According to 20 rules [23] described in Table I, the rule-based categorization module examines incoming requests to see if they are legitimate or malicious. The regulations consider the number of requests, packet size, SOAP size, nested attributes, entity count, DTD declaration in SOAP requests, and the overlong name of attributes or entities.

IP addresses are checked and compared to a list of IP addresses the firewall has already blacklisted. The firewall will reject the connection if the source IP address is on the blocked list. If the IP address is not one of the ones on the blacklist, the client request will be routed to the function that checks for request attempts, if necessary. Fig. 2 depicts the rule-based categorization module.

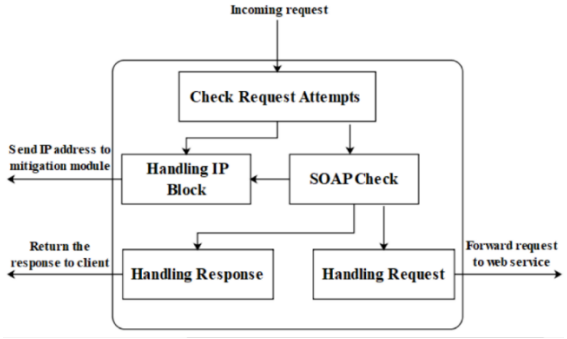

Figure 2. Classification module based on Rules.

To prevent flooding attacks, the middleware tool limits the number of requests. A client may make only one request during a given time. The default value is 1000 ms. For this purpose, customers' IP addresses and the most recent time they requested a web service are saved in a monitoring list. The total number of tries is calculated using the following formula: $T_t \leq (R_c - R_i)$ where:

T_t: Time threshold

R_c: Time of the current request

R_i: Time of the last request

TABLE I. SELECTION OF THE 20 RULES IN OUR WORK

No	Features								Decision
	No Request	No Invalid Request	Request Size	SOAP Size	Nested Attributes	Nested Entity	DTD Declaration	Overlong Name	
1	V	V	N	N	NO	NO	NO	NO	A
2	V	V	N	N	NO	NO	NO	YES	D
3	V	V	N	N	NO	NO	YES	NO	D
4	V	V	N	N	NO	YES	NO	NO	D
5	V	V	N	N	YES	NO	NO	NO	D
6	V	V	N	L	NO	NO	NO	NO	D
7	V	V	L	V	NO	NO	NO	NO	D
8	V	E	N	N	NO	NO	NO	NO	D
9	E	V	N	N	NO	NO	NO	NO	D
10	V	V	N	N	NO	NO	YES	YES	D
11	V	V	N	N	NO	YES	YES	YES	D
12	V	V	N	N	YES	NO	YES	YES	D
13	V	V	N	N	YES	YES	YES	YES	D
14	V	V	N	L	YES	YES	YES	YES	D
15	V	V	L	L	YES	YES	YES	YES	D
16	V	E	L	L	YES	YES	YES	YES	D
17	E	E	L	L	YES	YES	YES	YES	D
18	V	V	-	-	-	-	-	-	A
19	V	E	-	-	-	-	-	-	D
20	V	V	N	-	-	-	-	-	A

V= Valid; N = Normal; A= Allow; D= Deny; E= Exceed; L=Large.

The request's features are sifted through at this point. The features are tested and confirmed to ensure that the XML attack does not compromise the service. Following retrieving the characteristics, the decision is made using the 20 rules. Denied requests are discarded and routed to the Handling Response Function. The request will be routed to the Handling Request Function if the judge allows it. Table I shows a selection of the developed rules [23].

The current client's connection will be terminated, and the IP address of the client will be sent to the mitigation module.

To quickly remove the IP addresses of customers who make large requests to the web server, the mitigation module uses a block listing of suspicious IP addresses. Furthermore, before allowing or rejecting a client's request, the mitigation module compares the client's request ratio to the baseline ratio. In addition, the mitigation module can modify firewall rules to prevent future attacks.

The following HTTP response is sent to clients by the middleware tool see Table II, then, the web service receives the client's request and sends it to the Handling Request Function.

TABLE II. RETURNS RESPONSE WITH MEANING DETAILS

No	Return Response	Meaning Response
1	200	There is no problem with this request; it is OK
2	400	A request for an invalid request is a bad one
3	404	A method that is not authorized is marked with a 405 Method Not Allowed error.
4	405	Unable to locate the requested URL; not found
5	500	Internal Server Error, a Server error

IV. RESULT AND DISCUSSION

We will evaluate the recommended strategy based on the server response time. As a result, we use a regular mode to calculate the server response time for the pre-prepared SOAP queries. Fig. 3 depicts a typical mode of operation for the response time collection technique. Microsoft IIS version 10 security settings and ASMX web service countermeasures will be used in regular mode. Furthermore, the login page now includes a username and password field to improve security. There will be no assault in this situation.

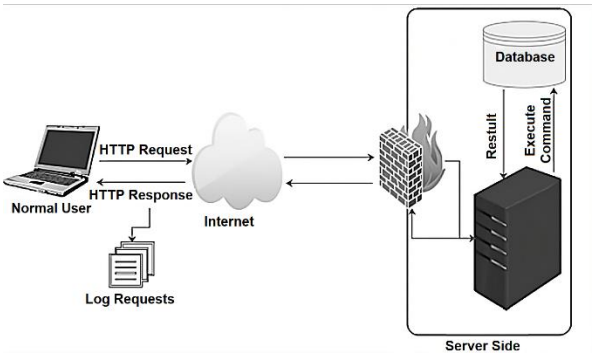

Figure 3. Time stamping in normal mode for server response times.

Scenario 1 experiment settings were identical to those in Normal mode. It was, however, simulated as if it were being attacked. By sending malicious SOAP queries to the web service, an attacker can cause a service outage. The SOAP requests generated by the typical user will be sent simultaneously. As a result, as shown in Fig. 4, each request would have to be analyzed.

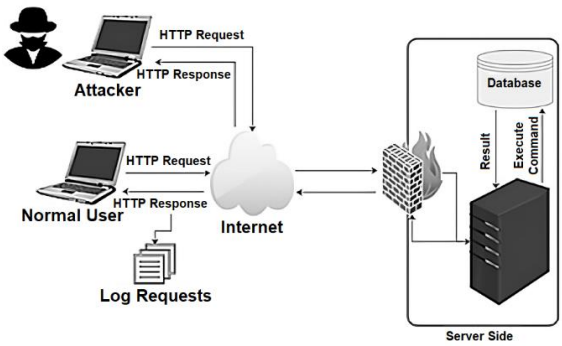

Figure 4. Scenario 1.

In Scenario 2, users could not access the web service directly from the system's back end. Instead, to access the web service, users had to utilize the middleware tool. The second case is shown in Fig. 5.

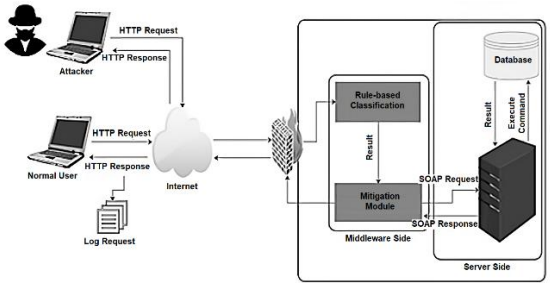

Figure 5. Scenario 2

The consequences of both situations will be discussed in this section. Based on the kind of XDoS assaults, the two scenarios are evaluated.

A. Dataset

In this section, we will attempt to determine the experimental controls. This investigation begins by employing the ASMX web service and the programming language. Net C#. Microsoft SQL 2014 shall serve as the database management system Version 10 of Internet Information Service (IIS) hosts the active website. This middleware product was developed using C# as its programming language. In addition, Windows Firewall is the add-on that enables network-layer security.

For this task, a cluster of four computers is utilized. The middleware server comprises two computers, one running Windows service provider and the other Windows Server 2016. Both of the other computers pose as legitimate users and malicious hackers.

The suggested technique is evaluated based on the server response time. Thus, the server response time for SOAP queries that have been prepared. The SOAP requests contain Normal, Oversized payload, Deep nested

payload, XML external entity, XML entity expansion, XML attribute count, XML entity count, and Overlong Names requests. The quantity per request is twenty.

The outcome is a draw based on the 20 rules listed in Table I. The 20 rules are evaluated using two situations in which each rule is under assault. The time that is recorded represents the server's response.

B. Attack with a Large Payload

Scenario 2 has outperformed Scenario 1 regarding the 20 "Oversized Payload" requests, as shown in Fig. 6, by employing the suggested defensive strategy. Scenario 1 has an average reaction time of 0.459 s, while Scenario 2 has an average reaction time of 0.006 s.

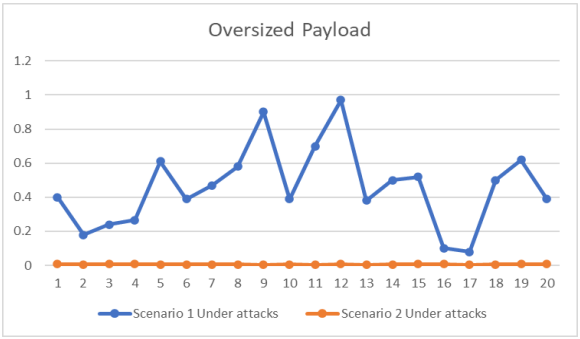

Figure 6. Result request "Oversized payload".

C. Attack with a Large Payload

The 20 'Deeply Nested Payload' requests show that Scenario 2 outperforms Scenario 1. Scenario 1 takes an average of 0.171 s to respond, whereas Scenario 2 takes only 0.007 s, as shown in Fig. 7.

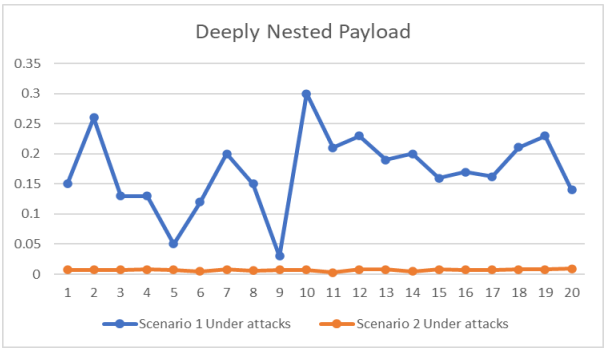

Figure 7. Result Request for deeply nested payload.

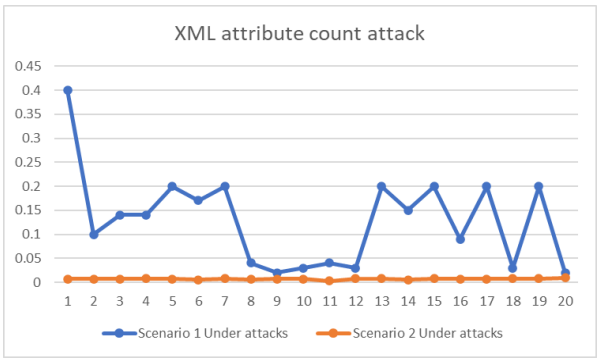

Figure 8. XML attribute count attack result.

D. Attack on the XML Attribute Count

Fig. 8 shows the results of both "XML attribute count attack" queries. According to the findings, Scenario 2 outperformed Scenario 1 for the "XML attribute count attack" queries. The average reaction time in Scenario 1 is 0.130 s, while the average response time in Scenario 2 is 0.007 s.

E. Attack on the XML Element Counts

Fig. 9 depicts the results of both "XML element count attack" requests. Scenario 2 outperformed Scenario 1 and the Normal scenario in this experiment, yielding the same results. Scenario 1 has an average response time of 0.118 s, while Scenario 2 has a response time of 0.006 s.

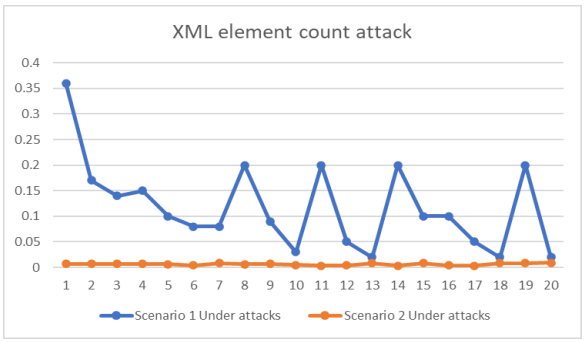

Figure 9. Response to the "XML element counts assault".

F. An Exploit against the XML Entity Expansion

Fig. 10 shows that Scenario 2 outperformed Scenario 1 for the twenty "XML entity expansion attack" requests, with an average response time of 0.067 s versus 0.126 s for Scenario 1.

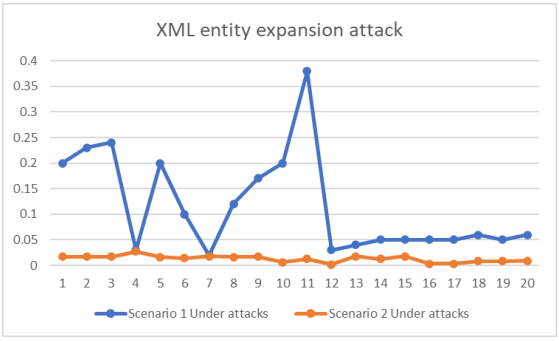

Figure 10. The 'XML entity expansion attack' request's outcome.

G. The Attack on an XML External Entity

Fig. 11 depicts the results of the "XML external entity attack" queries in both cases. Scenario 2 outperformed Scenario 1 for the twenty "XML external entity attack" queries, with an average response time of 0.067 s vs. 0.081 s.

H. Overlong Names Attack XML

Fig. 12 depicts the results of both scenarios for "overlong names" queries. Scenario 2 outperformed Scenario 1 for the twenty "XML external entity assault" requests. Scenario 1 has a response time of 0.085 s, while Scenario 2 has a response time of 0.070 s.

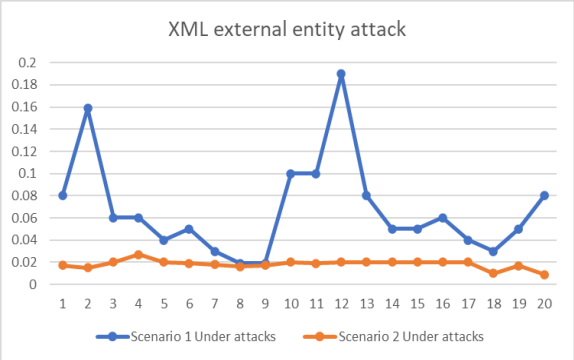

Figure 11. The attack request resulted in an "XML External Entity attack".

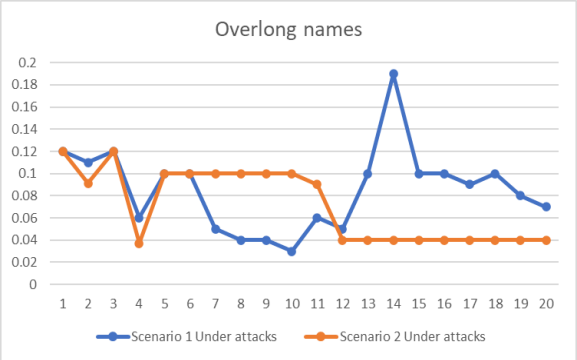

Figure 12. The attack request resulted in an "XML External Entity attack".

TABLE III. RESPONDING TIMES FOR THE SEVEN DIFFERENT KINDS OF SOAP QUERIES

Scenario No	Response of Time Average (ms)						
	Oversized payload	In deep payload	XML attribute	XML element	XML entity	XML external entity	XML overlong name
Sce.1	459	171	130	118	126	81	85
Sce.2	6	7	7	6	67	67	70

Table III summarizes the average comparisons of the seven XDoS assaults conducted on Scenario 1 (which did not use the proposed defense) and Scenario 2 (which did use the proposed defense).

The experiment was first run with the same parameters as the Normal setting. On the other hand, an attack simulation was staged to give the impression that the city was under siege. The attacker would send malicious SOAP requests to the web service for a denial-of-service attack. At the same time, the average user will submit the prefab SOAP requests. All demands would be weighed in this manner. This is illustrated in Fig. 4.

In this case, as shown in Scenario 2, the web service was in the backend and inaccessible to users. Users had to use the middleware tool to connect to the web service. Fig. 5 depicts the alternative case.

V. CONCLUSION

This study was conducted to learn more about popular web service frameworks and how they respond to denial-of-service attacks. Using relevant security research and web service attacking technologies, we built our Denial of Service (DoS) arsenal. These assaults were carried out after they had been planned. We devised a multi-phased testing strategy to assess how a service platform operates during attacks and detect potential attack effects. Using real-world methods ensures that application and service platform developers and providers can assess the security of their service platforms. The availability of web-based services is critical to ensuring business continuity. The consequences of XDoS attacks pose a severe threat to web-based systems. Massive, Distributed Denial of Service (DDoS) attacks necessitate significant computing resources, such as CPU and memory, rendering the system inaccessible to legitimate users. The XDoS attacks examined in this study demonstrated that they could halt a web server with limited resources.

A middleware tool has been designed and developed to address this issue. The tool's effectiveness in detecting and blocking these attacks has been demonstrated in testing. Furthermore, the middleware tool can detect and prevent Distributing Denial of Service (DDoS) and flooding attacks in near real-time. Furthermore, the middleware tool ensures that regular requests are serviced at a nearly 100% reliable rate. We will focus on time in the future, so we must devise a method to detect apps that can make bogus requests.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

The authors of this paper collaborated to present it in the most effective way possible, with AA proposing the methodology and designing the experiment. MAE formats the concept and writes and edits the paper; all authors have approved the final version.

REFERENCES

[1] B. S. Balaji, S. Balakrishnan, K. Venkatachalam, V. Jeyakrishnan *et al.*, "Automated query classification-based web service similarity technique using machine learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp. 6169–6180, 2021.

[2] Q. Yu, X. Liu, A. Bouguettaya, B. Medjahed *et al.*, "Deploying and managing web services: Issues, solutions, and directions," *The VLDB Journal*, vol. 17, no. 3, pp. 537–572, 2008.

[3] Z. Wu, Y. Yin, G. Li, M. Yue *et al.*, "Coherent detection of synchronous low-rate DoS attacks," *Security and Communication Networks*, vol. 2021, 2021.

[4] L. O'Brien, P. Merson, L. Bass *et al.*, "Quality attributes for service-oriented architectures," in *Proc. International Workshop on Systems Development in SOA Environments (SDSOA'07: ICSE Workshops 2007)*, IEEE, 2007, p. 3.

[5] R. Daigneau, *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and Restful Web Services*, Addison-Wesley, 2012.

[6] S. Bourekkache, O. Kazar, and A. Aloui, "Computer and network security: Ontological and multi-agent system for intrusion detection," *J. Digit. Inf. Manag.*, vol. 17, no. 3, p. 133, 2019.
[7] M. Jensen, N. Gruschka, R. Herkenhoner, N. Luttenberger *et al.*, "Soa and web services: New technologies, new standards-new attacks," in *Proc. Fifth European Conference on Web Services (ECOWS'07)*, IEEE, 2007, pp. 35–44.
[8] G. P. Bherde and M. Pund, "Recent attack prevention techniques in web service applications," *in Proc. 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, IEEE, 2016, pp. 1174–1180.
[9] M. Elhamam, A. Chillali, L. El-Fadil *et al.*, "Public key cryptosystem and binary Edwards curves on the ring F2n[e], $e^2 = e$ for data management," in *Proc. 2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, IEEE, 2022, pp. 1–4.
[10] B. Grobauer, T. Walloschek, E. Stocker *et al.*, "Understanding cloud computing vulnerabilities," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 50–57, 2010.
[11] N. Gonzalez *et al.*, "A quantitative analysis of current security concerns and solutions for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, pp. 1–18, 2012.
[12] K. Hashizume, D. G. Rosado, E. Fernández-Medina, E. B. Fernandez *et al.*, "An analysis of security issues for cloud computing," *Journal of internet services and applications*, vol. 4, no. 1, pp. 1–13, 2013.
[13] I. M. Khalil, A. Khreishah, M. Azeem *et al.*, "Cloud computing security: A survey," *Computers*, vol. 3, no. 1, pp. 1–35, 2014.
[14] M. Ali, S. U. Khan, A. V. Vasilakos *et al.*, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.
[15] M. Masdari and M. Jalali, "A survey and taxonomy of DoS attacks in cloud computing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3724–3751, 2016.
[16] O. Osanaiye, K.-K. R. Choo, M. Dlodlo *et al.*, "Distributed Denial of Service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016.
[17] S. Abdelsayed, D. Glimsholt, C. Leckie, S. Ryan, S. Shami *et al.*, "An efficient filter for denial-of-service bandwidth attacks," in *Proc. GLOBECOM'03, IEEE Global Telecommunications Conference*, IEEE, 2003, vol. 3, pp. 1353–1357.
[18] Y. Tao and S. Yu, "DDoS attack detection at local area networks using information theoretical metrics," in *Proc. 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, 2013, pp. 233–240.
[19] S. T. Zargar and J. Joshi, "DiCoDefense: Distributed collaborative defense against DDoS flooding attacks," in *Proc. IEEE Symposium on Security and Privacy*, Citeseer, 2013.
[20] E. Anitha and S. Malliga, "A packet marking approach to protect cloud environment against DDoS attacks," in *Proc. 2013 International Conference on Information Communication and Embedded Systems (ICICES)*, IEEE, 2013, pp. 367–370.
[21] C. Späth, C. Mainka, V. Mladenov, J. Schwenk *et al.*, "SoK: {XML} parser vulnerabilities," in *Proc. 10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.
[22] A. Alasri and R. Sulaiman, "Protection of XML-based denial-of-service and http flooding attacks in web services using the middleware tool," *International Journal of Engineering and Technology (UAE)*, vol. 7, no. 4, pp. 322–329, 2018.
[23] G.-Y. Chan, F.-F. Chua, C.-S. Lee *et al.*, "Intrusion detection and prevention of web service attacks for software as a service: Fuzzy association rules vs fuzzy associative patterns," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 2, pp. 749–764, 2016.