An Efficient Mechanism for handling CORS Vulnerabilities in Websites

Nishant Tomar
Department of Information Technology
Netaji Subhas University of Technology
110078, Delhi
nishanttomar21@gmail.com

S.K. Dhurandher
Department of Information Technology
Netaji Subhas University of Technology
110078, Delhi
dhurandher@nsut.ac.in

***ABSTRACT*—The Same Origin Policy places a "write-only" restriction on cross-origin network resource access. The Same Origin Policy places a "write-only" restriction on cross-origin network resource access. Many web applications, on the other hand, require "read" access to content from a different source. To get around the default Same Origin Policy constraint, developers have devised methods such as JSON-P. Security issues abound in such patchwork solutions. CORS (cross-origin resource sharing) is a web browser-supported method for addressing cross-origin network access. The results of our examination of CORS misconfigurations in the top 1 million Alexa websites and its solutions are presented in this paper. CORS design, development, and deployment revealed new security issues. In a handful of minor ways, CORS diminishes the cross-origin "write" privilege, which can be troublesome. In web interactions, CORS introduces new types of potentially harmful trust relationships. Developers misconfigure CORS due to its cryptic policy and numerous interconnections with other web protocols. Finally, we've implemented the CORS specification in key websites to solve security concerns.**

Keywords: CORS, XSS, CSRF, XSS, CAO, ACAC, Penetration Testing, Exploit, Security, Vulnerabilities, Solution

I. INTRODUCTION

CORS is a client-side web security protocol that addresses the same-origin principle (SOP). It prevents scripts from accessing web resources from another source. The default SOP for sharing cross-origin network resources does not include an explicit access control authorization procedure [1]. Because browsers don't support cross-origin network resources, developers developed an ad-hoc workaround. JSON uses cross-origin JavaScript to get around the ban. However, such a workaround solution comes with security risks. CORS is a protocol that allows for controlled access to cross-origin network resources [2]. It is offered as a solution to the P's of JSON and as a protocol for allowing authorized access to cross-origin network resources. Since 2009, major browsers (such as Chrome, Firefox, and Internet Explorer) have supported this protocol, and it is now a webpage staple. Our research analyses CORS' architecture, implementation, and deployment for security flaws, as well as expose novel CORS security flaws in real-world websites. The issues discovered during our investigation can be divided into two categories:

a) Excessively broad permits for cross-origin transmission. Inadvertently, the CORS protocol provides for higher default transmission permissions, giving attackers more power and introducing new security risks. By utilizing this relaxed sending authorization. An attacker could exploit previously unknown CSRF vulnerabilities, remotely deduce a victim's cookie size or internal binary protocol services.

b) CORS security flaws. CORS requires resource servers to trust third-party domains. Third-party website use expands the attack surface and exposes users to new security concerns. CORS is a straightforward concept, but there are a few nuances that can lead to a host of misconfigurations and security problems in the real world if they are overlooked. According to Alexa's top 50,000 websites, there are 132,476 sub-domains with insecure CORS misconfigurations, making up 27.5 percent of all sub-domains with CORS configurations on 13.2 percent of all "base domains" with CORS setups ("public suffix plus one" refers to the initial public domain suffixes."). You'll see them on websites such as sogou.com, FedEx, and the Washington Post, amongst others [3].

Data theft, account fraud, and even identity theft might all be the outcome of these misconfigurations looking at several famous CORS implementations to determine the difficulties' real-world impact [4]. The paper used a large security firm's open-to-researchers passive DNS information to extract Alexa's Top 50,000 subdomains. 97,199,966 subdomains were identified across 49,729 base domains. In separate testing requests, modification on the Origin header value for each subdomain to various error-prone values and determined their CORS installations based on response headers.

A. Contribution

The contribution to this paper is founding several CORS related misconfigurations in Alexa's top 1 million websites and hence gave a solution to solve all those problems which can later lead up to vulnerabilities which can be exploited, with 100% efficiency. Running our proposed solution or algorithm at an interface or gateway between server-side and client-side while the response is coming from the server side. Running the solution at the interface level and sent back the solution to further mitigate the problem.

This paper is organized as follows. Section 2 discusses the related work done in this field with reference of top 10 research paper published in this area. Section 3 considers the system model proposed by us and section 4 discusses the implementation and evaluation. Section 5 comprises of the conclusion and future scope to this problem.

II. RELATED WORK

This research examines the composition and evolution of CORS, as well as the application of CORS in open-pit mine surveying, deformation monitoring, and vehicle scheduling. The CORS system's foundation will increase total open pit mine surveying accuracy and efficiency, reduce surveying

costs, boost accuracy and automation of deformation monitoring, and improve operating efficiency of the vehicle scheduling system through analysis and comparison. Mining's digital transformation has been accelerated because to CORS's establishment, which has helped the industry grow while also enhancing the operation's management and output efficiency. As a result, major open pit miners across the country need to build their own CORS systems. The limitation is that the paper only discusses the problem in theory and there is no proposed work given to his theory [6].

It is inevitable that data that is shared online will be exposed as a result of ever-changing technologies. Using the Wireshark tool, an ethical hacker can discover the security issues in the system's user authentication scheme. If you're looking for a fast and effective way to detect security flaws, this testing procedure is the best option available. When Wireshark is used, you can rest assured that the technique you are following is up to snuff as well. Wireshark's advantages in penetration testing are covered in this article, as is a way for employing the tool that is illustrated in the paper. The vast majority of a network's components are vulnerable to intruders' attempts to compromise its security. This research aims to address the aforementioned problem by doing a survey of the various penetration testing tools on the market. Wireshark penetration testing is demonstrated in this example as well. The limitation is that the paper only assess the problem in using a tool called Wireshark and there is no proposed work provided [7].

As a result of the default Same Origin Policy, cross-origin network resources are effectively "write-only." Many web applications, on the other hand, call for "In order to avoid the default Same Origin Policy restriction, developers have come up with workarounds, such as JSON-P. Ad-hoc solutions like these put a slew of security holes in place. To handle cross-origin network access, all major web browsers implement CORS (cross-origin resource sharing), a more strict approach. In this paper, describe the results of an empirical investigation into the practical applications of CORS. The design, development, and deployment of CORS face a variety of additional security concerns. The cross-origin restrictions of CORS are loosened "CORS introduces new forms of risky trust dependencies into web interactions; 2) CORS is generally not well understood by developers, possibly due to its inexpressive policy and its complex and subtle interactions with other web mechanisms, leading to various misconfigurations. After studying the security flaws discovered, paper recommend a series of protocol changes and clarifications. CORS and major browsers have both incorporated a few of our ideas. The limitation is that the paper only discusses the problem in theory and there is no proposed work given to his theory [8].

A web application built with HTML5 will have a significant advantage over those built with earlier versions. Some difficulties can be solved by utilising cookies alone, but the Local-Storage feature introduces new security risks due of the use of CORS. There are numerous cross-domain techniques that can be used to store data in Local-Storage, however this paper focuses on the present security vulnerabilities in CORS. Test cases are created to help researchers better understand how to safeguard the privacy of their users' personal information and to present a workable plan for implementing CORS. It's concluded that CORS can be used, and several recommendations are made for improving CORS security. Limitation of the paper is that a solution is prosed but there is no validation for that solution [9].

Precisely because it involves "getting into your own system to see how difficult it is to do so," red teaming is often referred to as ethical hacking or penetration testing by students. Penetration testing, in contrast to popular belief, necessitates an in-depth examination of potential risks and attackers. Penetration testing data must be interpreted correctly to be useful. If a penetration test fails to uncover any problems, it does not mean that the system is secure, nor does it mean that no faults exist. The absence of faults cannot be detected by any sort of testing. Testing can only tell us that the faults they were looking for and failing to find aren't there, which gives us an indication of the overall security of the design and execution. The limitation is that the paper only discusses the problem in theory and there is no proposed work given to his theory [10].

Petri-nets can be a valuable tool for modelling penetration testing. New benefits have been added while maintaining some of the strengths of the defect hypothesis and attack tree methodologies. The limitation is that the paper only discusses the problem in theory and there is no proposed work given to his theory [11].

The paper is entrusted with verifying that a software application meets its functional business requirements, which is a broad goal for quality assurance and testing organizations. Typically, dynamic functional testing is used to check that the application's features are properly implemented. Although security is not a feature or even a group of features, security testing does not suit this paradigm. The limitation of this paper is that it uses a already made software for penetration testing and there is no proposed solution to the problem in hand [12].

In addition to highlighting security flaws, penetration testing helps to secure networks. Penetration testing tools, attack methods, and defense measures are all examined in this article. Specifically, private networks, devices, and virtualized systems and tools to conduct a variety of penetration tests. The majority of our work was done with the Kali Linux set of tools. The attacks were performed included: smartphone penetration testing, hacking phones Bluetooth, traffic sniffing, hacking WPA Protected Wi-Fi, Man-in-the-Middle attack, spying (accessing a PC microphone), hacking phones Bluetooth, and hacking remote PC via IP and open ports using advanced port scanner. After that, the findings are distilled and discussed. In addition, the report provided extensive instructions on how to carry out these attacks. The limitation is that the paper only discusses the problem in theory and there is no proposed work given to his theory [13].

As a defensive measure, Intrusion Detection Systems (IDS) look for online security threats. Internet security systems (IDS) are well-known for detecting network-based assaults, but their ability to monitor and identify web-based attacks is still in its infancy. According to this study, a web-based application-specific IDS design methodology should be

presented. This paper discusses a number of specific characteristics that complicate the monitoring and detection of web attacks by an IDS. Also covered in depth in the study is the current state of the art in terms of web traffic monitoring detecting systems. The IDS can be compared in several ways, based on their design and functionality, using a variety of dimensions. In addition, the paper provide a conceptual framework for a web IDS with a preventative mechanism that provides systematic direction for the system's deployment. AppSensor, AQTRONIX WebKnight, AQTRONIX WebKnight, ModSecurity, and Shadow Daemon are all compared. The cutting-edge facts in this article will greatly assist interest groups in identifying the domain's strongest and weakest areas, laying a solid foundation for the creation of an intelligent and efficient system. The limitation is that the paper only proposes the issue of intrusion detection in theory and there is no proposed work given to his theory [14].

Users can perform remote operations and transport any type of data via XML-based SOAP Web Services, which are widely utilised. Modern applications include SOA, cloud interfaces, federated identity management, eGovernance and military services. As a result of this technology's widespread usage, a large number of often-complex extension specifications have emerged. Because of that, Web Services assaults have been increasing at an alarming rate. In addition to a specific Denial of Service attack, they include attacks on cloud service providers' interfaces and the confidentiality of encrypted communications. In order to assess the security of their web applications, developers use a variety of penetration testing techniques. Penetration testing tools for Web Services-specific attacks do not yet exist, in contrast to well-known techniques such as SQL injection and Cross-Site Scripting. First, WS-Attacker, the first automated web service security penetration testing tool, was developed. WS-Addressing spoofing and SOAP-Action spoofing attacks are discussed and evaluated in this article, as are our design decisions and the resistance of four Web Services frameworks to these assaults. The limitation of this paper is that it uses pen-testing tools for penetration testing and there is no proposed solution to the problem [15].

III. SYSTEM MODEL

A. Motivation

The motivation towards writing this paper is that it saw a lot of research paper talk about the theoretical aspects of CORS configurations and misconfigurations along with its side-effects but no paper suggests any solution to those issues along with validation to there solutions. With the help of this paper, website developers can have an effective mechanism to solve CORS misconfigurations and in-turn not let XSS, CSRF etc. cyber attacks on the websites due to CORS misconfigurations.

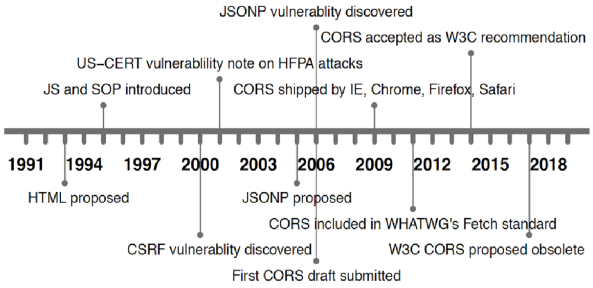

Figure 1: CORS timeline for access to the network

B. List of problems identified

Misconfiguration of CORS (Cross-Origin Resource Sharing) is an intentional technique for exposing flaws in the Same-Origin Policy (SOP). The ACAO header can be used by web servers to allow cross-site access. The Origin browser header, for example, generates its value dynamically based on user input. Misconfigured resources can be accessed by unwanted websites. If Access-Control-Allow-Credentials (ACAC) is set, an attacker could access a logged-in user's sensitive information. This vulnerability is nearly as hazardous as cross-site scripting. Configurations of the CORS:

Misconfiguration	Description
Developer backdoor	Insecure developer/debug origins like JSFiddler CodePen are allowed to access the resource
Origin reflection	The origin is simply echoed in ACAO header, any site is allowed to access the resource
Null misconfiguration	Any site is allowed access by forcing the null origin via a sandboxed iframe
Pre-domain wildcard	**notdomain.com** is allowed access, which can simply be registered by the attacker
Post-domain wildcard	**domain.com.evil.com** is allowed access, can be simply be set up by the attacker
Subdomains allowed	**sub.domain.com** allowed access, exploitable if the attacker finds XSS in any subdomain
Non-SSL sites allowed	An HTTP origin is allowed access to a HTTPS resource, allows MitM to break encryption
Invalid CORS header	Wrong use of wildcard or multiple origins, not a security problem but should be fixed

Table 1: CORS misconfigurations and its description.

C. Proposed Solution

CORS attacks can be prevented considering taking actions Such as the following:

1) Specify the allowed origins

The Access-Control-Allow-Origin header should specify the permissible origins in full if a web resource includes sensitive information (i.e., no wildcards).

2) Allow only trusted sites

This may seem apparent, considering the prior suggestion, but origins listed in the Access-Control-Allow-Origin header should only come from trusted sites. In other words, unless the website is a public API endpoint with no authentication requirements, you should avoid dynamically reflecting origins from cross-domain request headers without validation.

3) Don't whitelist "null"

The Access-Control-Allow-Origin: null header should be avoided. However, even if internal documents and sandboxed

requests might specify the "null" origin, you should consider cross-domain cross-origin requests the same way you treat external cross-domain requests. Your CORS headers need to be defined correctly.

4) Implement proper server-side security policies

A secure web server is more than just properly configured CORS headers. It's one of the components, but it's not the whole story. When it comes to safeguarding sensitive data, CORS is never a replacement for the measures taken on the server side. In addition to properly configuring CORS, you should maintain securing sensitive data, such as authentication and session management. To avoid being a victim of a CORS attack, it's critical for you to stay one step ahead of phishing scams and malicious websites and downloads. Tips like this can be helpful. It's a good idea to follow these procedures because they apply to a wide range of online threats, including avoiding bogus antivirus [16].

- Make use of an inbound firewall - Every major operating system and commercial router on the market has an inbound NAT firewall built in. It's a good idea to turn them on just in case you accidentally click on a dangerous link.
- Ensure that the antivirus software you purchase from a reputable source is authentic and that it is configured to run regular checks.
- Pop-ups should never be clicked on - There is no telling where they're going to take you next.
- If your browser warns you about a website you're trying to access, you should look elsewhere for the information you need [17].
- You should never open an attachment in an email unless you are sure who sent it and what it contains.
- Unless you know exactly who gave the link and where it leads, don't click on links in emails (URLs). Finally, take a close look at the link. What kind of link is it? Today, HTTPS is used by the vast majority of reputable websites. Is the link spelled incorrectly (Facebook instead of Facebook)? Instead of using the link, if you can go to your destination without it, do so [18].

The time and space complexity of proposed algorithm and solution is O(n).

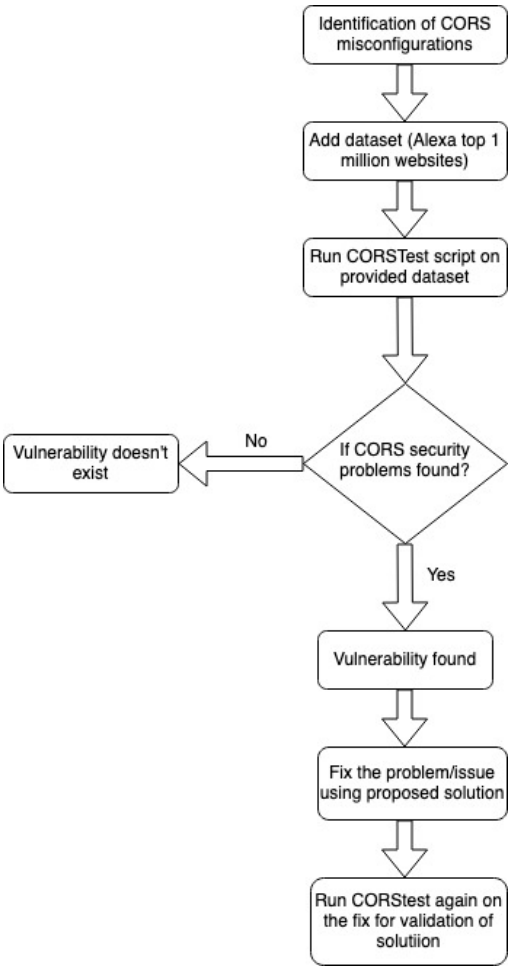

Figure 2: Flow diagram of CORStest tool and its solution along with solutions validation.

IV. IMPLEMENTATION AND EVALUATION

CORStest is a Python-based CORS misconfiguration detector that offers parallelization. It checks a text file for misconfigured domain names or URLs.

To find vulnerabilities, CORStest examines origin request headers for the presence of access-control-allow-origin Run the CORS tester to find vulnerabilities. For research the algorithm ran CORStest on Alexa's top 1 million sites to see how many have wide-open CORS setups.

A. Experimental Setup

The system on which the test and its solution ran was MacOS Monterey with processor 1.6 GHz LPDDR3 and RAM 8 GB 2133 MHz LPDDR3. The network provider was Excitel with Ping – 5 ms, Download speed – 116.87 Mbps, Upload speed – 137.74 Mbps. On this network, the test ran rigorously for 14 hours with multiprocessing with pool size 32 running concurrently. There was no other work done on the system till the test wasn't complete.

B. Dataset

CORStest was run on the Alexa top 1 million sites to determine how many sites have wide-open CORS settings on a bigger scale: Commands for copying the dataset and evaluating the website for CORS misconfiguration vulnerabilities [19].

```
$ wget -q http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
$ unzip top-1m.csv.zip
$ awk -F, '{print $2}' top-1m.csv > alexa.txt
```

These websites are evaluated based on CORS configuration and the vulnerabilities are found on websites as a result of misconfgurations of ACAO ad ACAC headers in CORS. The dataset comprises of top 1 million Alexa searched websites to test the configuration settings and find the vulnerabilities within. These websites are cloned with the help of the above commands and CORS misconfiguration is discovered [20].

```
usage: corstest.py [arguments] infile

positional arguments:
  infile        File with domain or URL list

optional arguments:
  -h, --help     show this help message and exit
  -c name=value  Send cookie with all requests
  -p processes   multiprocessing (default: 32)
  -s             always force ssl/tls requests
  -q             quiet, allow-credentials only
  -v             produce a more verbose output
```

Figure 3: Configurations possible in CORS tester for filtration.

The infile contains list of top one million allexa searched websites. File has different arguments acting as filters which are helpful to display the output differently.

C. Results

1. CORStest Results

On the normal connection took about 14 hours to run the CORS misconfiguration.

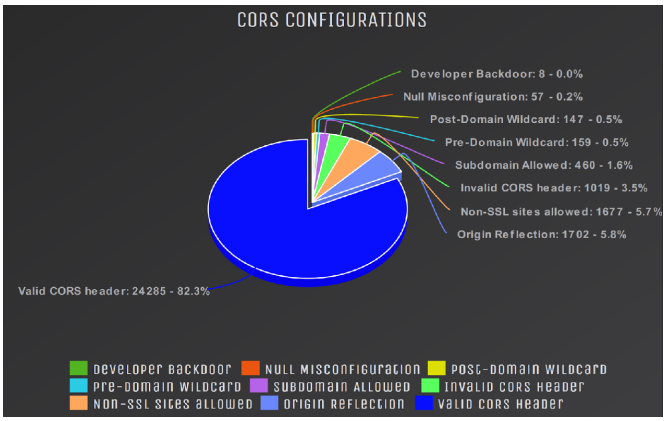

Figure 4: CORS misconfigurations found in 1 million websites.

S.No.	Misconfigurations	No. of websites	Percentage of websites
1	Developer backdoor	8	~ 0.0
2	Origin reflection	1702	5.8
3	Null misconfiguration	57	0.2
4	Pre-domain wildcard	159	0.5
5	Post-domain wildcard	147	0.5
6	Subdomains allowed	460	1.6
7	Non-ssl sites allowed	1677	5.7
8	Invalid CORS header	1019	3.5
9	Valid CORS header	24285	82.3

Table 2: CORS misconfigurations found in 1 million websites.

Only 29,514 websites (3%) have CORS on their homepage (i.e., they responded with Access-Control-Allow-Origin). Google, for example, only activates CORS headers for certain URLs. The research might have crawled certain URLs (including subdomains) and used CORStest. Using a longer process would have been inefficient; our quick one should suffice. As a result, no CORS preflight was used in the test, which focused solely on HTTP:// requests (with redirects followed) [9]. Be aware that a website using the origin header does not necessarily guarantee that it is correct in all cases. The context is crucial; this configuration is ideal for public sites or API endpoints that are accessible to the public. Payment and social media sites may suffer. ACAC: True (ACAC) must also be set to enable access control. The study conducted more tests at additional locations that returned this header (see CORStest –q).

Nearly every website that accepts ACAO and ACAC headers has CORS misconfigurations attackers may use vulnerabilities through backdoor and reflection, misconfiguration by pre/post domain wildcard:

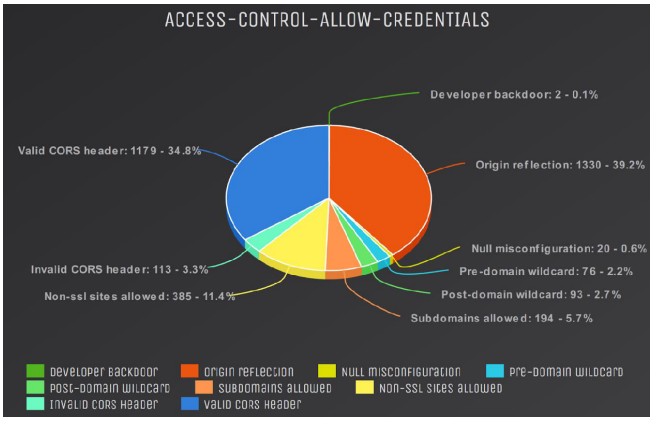

Figure 5: Access control allow credentials misconfiguration found in 1 million websites.

S.No.	Misconfigurations	No. of websites	Percentage of websites
1	Developer backdoor	2	0.1
2	Origin reflection	1330	39.2
3	Null misconfiguration	20	0.6
4	Pre-domain wildcard	76	2.2
5	Post-domain wildcard	93	2.7
6	Subdomains allowed	194	5.7
7	Non-ssl sites allowed	385	11.4
8	Invalid CORS header	113	3.3
9	Valid CORS header	1179	34.8

Table 3: Access control allow credentials misconfiguration found in 1 million websites.

2. Proposed Solution Results

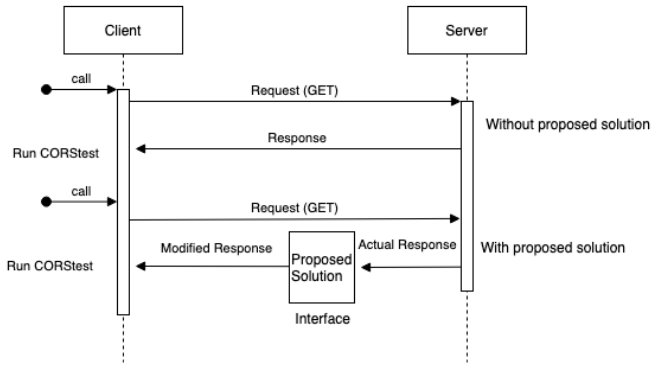

Figure 6: Proposed and non-proposed solution sequence diagram

When the response comes from the server side, algorithm iis run at the interface level between server and client side. The result of the algorithm or proposed solution is as follows:

S.No.	Misconfigurations	No. of websites without proposed solution	No. of websites with proposed solution
1	Developer backdoor	8	0
2	Origin reflection	1702	0
3	Null misconfiguration	57	0
4	Pre-domain wildcard	159	0
5	Post-domain wildcard	147	0
6	Subdomains allowed	460	0
7	Non-ssl sites allowed	1677	0
8	Invalid CORS header	1019	0
9	Valid CORS header	24385	0

Table 4: CORS misconfigurations found in 1 million websites after running proposed solution.

S.No.	Misconfigurations	No. of websites without proposed solution	No. of websites with proposed solution
1	Developer backdoor	2	0
2	Origin reflection	1330	0
3	Null misconfiguration	20	0
4	Pre-domain wildcard	76	0
5	Post-domain wildcard	93	0
6	Subdomains allowed	194	0
7	Non-ssl sites allowed	385	0
8	Invalid CORS header	113	0
9	Valid CORS header	1179	0

Table 5: Access control allow credentials misconfiguration found in 1 million websites after running proposed solution.

Therefore, it can be said that our proposed algorithm is 100% efficient in mitigating the cyber-attacks caused because of CORS misconfigurations.

D. Observations

• Various websites employ faulty CORS headers and ACAO headers with numerous origins. Domain, origin, SAMEORIGIN, self, true, false, undefined, None, 0, (null)
• Rack: CORS maps origins "or origins '*' into arbitrary origins; this is dangerous since developers may think "allows nothing and '*' operates according to the spec: Developers may think "allows nothing and '*' works".
• The problem appears to be caused by erroneous advice on the Internet rather than a flaw in IIS (Internet Information Service).
• Nginx delivers websites with origin reflections better than Phusion Passenger, although this is owing to dangerous configs stolen from Stackoverflow.
• The null ACAO value may be based on computer languages that return null if no value is supplied. The book provides code like var origin Whitelist = ['null',...], which developers could misunderstand as safe.
• CrVCL PHP Framework inserts ACAC and ACAO headers if CORS is enabled. Sub.domain.com.evil.com has post-domain and pre-sub-domain vulnerabilities.
• Scam sites using the words "Solo Build it!" http://sbiapps.sitesell.com/Access-Control-Allow-Origin is the answer. Some sites, such as / or /, have fixed ACAO values. What should browsers do in this situation? At the very least, inconsistency rules! Although Internet Explorer and Edge do not accept arbitrary sources, Firefox, Chrome, Safari, and Opera do.

V. CONCLUSION AND FUTURE SCOPE

A. Conclusion

This paper started with finding enormous vulnerabilities from the dataset mentioned and then further the eradication of all those vulnerabilities are done to make the website safe from attackers**.** Finally, it was proposed several adjustments and clarifications to address these issues. Cross-origin transmitting authorization in the usual SOP already offers major security risks, including CSRF and HFPA attacks. This permission allows messages to be sent in an abnormally loose manner. CORS could have handled all cross-origin access if backward compatibility had not been taken into consideration. This would have stopped CSRF, HFPA, and other cross-origin attacks. CORS maintains the former policy's backward compatibility. The research contains some limitations that there can be more efficient method to find vulnerabilities in website so that data can be accumulated in less time then propose in this research paper.

B. Future Scope

1. The future scope to this paper is that the CORStest and its solution with validation would prove to be extremely beneficial in identifying future avenues in the research domain of penetration testing in web applications.
2. There can be a tool or software made that can detect CORS misconfiguration and automatically configure it so that there would be no exploitation because of it.
3. There can be a live tool made that will trace if there is any live hacking taking place in the server and the tool stops that from happening.

REFERENCES

[1] Anley, C.; Heasman, J.; Lindner, F. and Richarte, G. The Shellcoder's Handbook: Discovering and Exploiting Security Holes. 2007. Wiley.
[2] Hayajneh, T.; Almashaqbeh, G.; Ullah, S. A Green Approach for Selfish Misbehavior Detection in 802.11-Based Wireless Networks. Mobile Netw. Appl. 2015, 20, 623–635.

[3] Panyim, K.; Hayajneh, T.; Krishnamurthy, P.; Tipper, D. On limitedrange strategic/random jamming attacks in wireless ad hoc networks. In Proceedings of the IEEE 34th Conference on Local
[4] Hayajneh, T.; Krishnamurthy, P.; Tipper, D.; Le, A. Secure neighborhood creation in wireless ad hoc networks using hop count discrepancies. Mobile Netw. Appl. 2012, 17, 415–430.
[5] Hayajneh, T.; Krishnamurthy, P.; Tipper, D. Deworm: A simple protocol to detect wormhole attacks in wireless ad hoc networks. In Proceedings of the IEEE 3rd International Conference on Network and System Security, Gold Coast, Australia, 19–21 October 2009; pp. 73–80.
[6] H. -J. Dong, A. -G. Xu, Z. -C. Tian and L. Gao, "Research on Application of CORS in Open-Pit Mine," 2012 2nd International Conference on Remote Sensing, Environment and Transportation Engineering, 2012, pp. 1-3, doi: 10.1109/RSETE.2012.6260691.
[7] S. Sandhya, S. Purkayastha, E. Joshua and A. Deep, "Assessment of website security by penetration testing using Wireshark," 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 2017, pp. 1-4, doi: 10.1109/ICACCS.2017.8014711.
[8] Jianjun Chen, Jian Jiang, Haixin Duan, Tao Wan, Shuo Chen, Vern Paxson, and Min Yang. 2018.Still don't have secure cross-domain requests: an empirical study of CORS. In Proceedings of the 27th USENIX Conference on Security Symposium (SEC'18). USENIX Association, USA, 1079–1093.
[9] N. Zhu, "Security of CORS on LocalStorage," 2021 International Conference on Internet, Education and Information Technology (IEIT), 2021, pp. 141-146, doi: 10.1109/IEIT53597.2021.00038.
[10] M. Bishop, "About Penetration Testing," in IEEE Security & Privacy, vol. 5, no. 6, pp. 84-87, Nov.-Dec. 2007, doi: 10.1109/MSP.2007.159.
[11] McDermott, James P. "Attack net penetration testing." In Proceedings of the 2000 workshop on New security paradigms, pp. 15-21. 2001.
[12] B. Arkin, S. Stender and G. McGraw, "Software penetration testing," in IEEE Security & Privacy, vol. 3, no. 1, pp. 84-87, Jan.-Feb. 2005, doi: 10.1109/MSP.2005.23.
[13] M. Denis, C. Zena and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2016, pp. 1-6, doi: 10.1109/LISAT.2016.7494156.
[14] Nancy Agarwal, Syed Zeeshan Hussain, and Vincenzo Conti. 2018. A Closer Look at Intrusion Detection System for Web Applications. Sec. and Commun. Netw. 2018 (2018).
[15] C. Mainka, J. Somorovsky and J. Schwenk, "Penetration Testing Tool for Web Services Security," 2012 IEEE Eighth World Congress on Services, 2012, pp. 163-170, doi: 10.1109/SERVICES.2012.7.
[16] T. Hayajneh, BJ Mohd, A. Itradat, AN Quttoum “Performance and Information Security Evaluation with Firewalls,” International Journal of Security and Its Applications, SERSC, Vol. 7, No. 6, pp 355-372, 2013. (DOI: 10.14257/ijsia.2013.7.6.36)
[17] Bassam J. Mohd, Thaier Hayajneh and Athanasios V.Vasilakos, A Survey on Lightweight Block Ciphers for Low-Resource Devices: Comparative Study and Open Issues, Journal of Network and Computer Applications, doi: 10.1016/j.jnca.2015.09.001.
[18] Bassam J. Mohd, Thaier Hayajneh and Athanasios V.Vasilakos, A Survey on Lightweight Block Ciphers for Low-Resource Devices: Comparative Study and Open Issues, Journal of Network and Computer Applications,
doi: 10.1016/j.jnca.2015.09.001
[19] Panyim, K.; Hayajneh, T.; Krishnamurthy, P.; Tipper, D. On limitedrange strategic/random jamming attacks in wireless ad hoc networks. In Proceedings of the IEEE 34th Conference on Local Computer Networks, Zurich, Switzerland, 20–23 October 2009; pp. 922–929.
[20] SINGH, K., MOSHCHUK, A., WANG, H. J., AND LEE, W. On the incoherencies in web browser access control policies. In Security and Privacy (SP), 2010 IEEE Symposium on (2010), IEEE, pp. 463–478.