International Journal of Engineering & Technology, 7 (2.3) (2018) 9-13

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

API vulnerabilities: current status and dependencies

Touhid Bhuiyan, Afsana Begum*, Sharifur Rahman, Imran Hadid

Dept. of Software Engineering, Daffodil International University, Dhaka, Bangladesh
Corresponding author E-mail: [2]afsana.swe@diu.edu.bd

Abstract

Recently API (Application Programming Interface) is becoming more popular for developers. When software is designed, most of the time, developers need to use APIs to manage a specific task. Developers use various kinds of APIs. Some of them are built by themselves and some are used from public APIs. API is a set of functions and procedures that allows another program or application to get access to features or data. Public APIs are open in public networks; developers collect these APIs depending on their specific needs. Developers need to interact with other software, as a result, a developer can conduct specific task without authorization to access the entirety of the software. It definitely reduces our loads at the same time introduces risks. In the end every developer wants to ensure security to his/her application. Commonly used public APIs are not enough secure to provide security to confidential data. We focused on these public APIs that are commonly used by developers. We tested a set of public APIs in our security lab and we have found many vulnerabilities that are highly alarming for developers who are going to use these API. In this paper we have tried to introduce the current status of vulnerable APIs. Moreover, several relationships exist between API vulnerabilities. In this paper we have also discussed the dependencies and relationships between API vulnerabilities.

Keywords: *API; API Security; Vulnerability; Public API's; API Vulnerability; Test API vulnerabilities; API IDOR; API CORS; API Problems;*

1. Introduction

API is a program or system that is accessible by other programs. API has brought revolutionary change in present applications, programs or systems. Developers, especially startup developers, use public APIs for their startup businesses to earn better revenue. However, API has some vulnerability that act as a threat to confidential data. For API owners ensuring proper security for their stake holders is not a priority. APIs are providing various types of services. A system uses API by providing an interface between them, API is not a database but it can perform better than a database. A database works with its own database language whereas API communicates with applications using protocol. API acceptance is that API provides amongst to the end users, because of the easy accessibility to a system. If API usability wants to increase first of all we have to think about facility of a developers and overall system security. For beginners, a common issue regarding API is the level of difficulty. In some cases, incorrectness leads to Vulnerabilities and has a large security impact. To reduce these problems, API designers doing huge mistakes, they trying to make simple design and provide simplicity to use in other application. Now it's also helpful for an attacker, so only a developer's perspective is insufficient if you want to increase API usability. Security has become a major necessity. If this is not ensured, API usability will lose widespread acceptance.

Apple phone IOS uses app review services that detect private API which contains sensitive data. A harmful program can steal data from another program due to the absence of sufficient security protocol. Randomly we collect API for our testing by using Google search and from testing platforms. Public APIs are commonly used by startup developers but these APIs are more risky to use in a system. We do list out public API's and the current security layer that they provide. Currently, most of the APIs are vulnerable to various kinds of vulnerability. Maximum API's are responsible for more than single vulnerabilities. Testing API vulnerability is more important than testing a system. Moreover, developers should test all of those. It is a necessity to be through so that APIs can securely communicate with clients. Although API owner uses some security protocol, however these securities layer are insufficient in providing proper security. That's why developers should be more concerned about all possible vulnerabilities. We are going to elaborate that these are not enough for security purposes to enhance use of public API.

2. Literature Review

We can see that there are a lot of cyber-attacks happening in every day. We can see some statistics about web attacks [18], [19], [20], [21] . Some vulnerability gives access to a database; some provides useful data from a database, some can inject or modify source file of a web application [22], [20], [21]. Some vulnerability focuses on crypto graphical retrieval or modification [23], [24]. As our main focus is on public API, we studied on API vulnerabilities. API has various types of vulnerability, and various types of tools exist at present. HackerTarget.com has 12 piece scanners to test API, but some researcher proposes new security

testing tools for VAS (vulnerability Assessment System) [1]. Some papers propose ways of increasing API usability [2]. But API usability will automatically improve if security issues can be resolved. Some paper propose new security layer for a specific device only but they didn't talk about API design policy [3],[10],[14]. Any security issues network, computers, servers or databases are responsible for representing security risks that can be exploited by hackers to gain access to system information [4]. JavaScript to Java interface vulnerability that allows entrusted JavaScript which via sandbox can allow remote code execution and it causes root access in Android phones [5]. For irregular kernel interface especially in Linux, LXFI can introduce privilege escalation vulnerability [6]. API's have common vulnerability that we have shown in the introduction [7]. Public API should have their own API documentation, which helps developers. API is not only helpful for developers but it also makes user access easier and reduces tasks [8]. We accessed various tools and links [9],[11],[12],[13],[15],[16],[17] to the rules to testing, finding APIs and related information. We have studied some challenges, issues and new technique to enhance security [25], [26] also.

3. Methodology

We have targeted public API which is collected from the open internet by using Google search and some are collected from authorized security testing platform [11] [17]. We tested those samples API using manually and automatically. We tested one by one by checking commonly available vulnerability. But software fault that causes kernel isolation[10] is a bit different from our working methodology. We covered web public API [12] which is responsible for developing new systems and includes their faults.

3.1 Plan

First of all, we are going to demonstrate API's current vulnerabilities and possible vulnerabilities. We collected similar information from several security research platform and reports [16] [13]. IDOR, CORS, RATE-LIMIT, Broken-authentication and RCE are critical vulnerability which exists in public API, moreover CSRF, ClickJacking also affects API. Our plan is to test some APIs and find out whether any vulnerability exists or not. If any vulnerability is present then for the existent vulnerability, is there any possibility to find vulnerability. Then we will try to find out how much access we get for the existing one vulnerability and show some result of our study.

a) Goal:

From tested sample API we want to come to a decision to determine the level of impact of vulnerabilities and the percentage of API that is vulnerable each. After that, we research current security systems that those APIs contain. Then we will provide countermeasures against those vulnerabilities.

b) Scope:

From existing research and gathering knowledge [15] we fixed our scope range. And we avoided handling RCE testing and API integrity that relate to the kernel module. We used the black box and white box testing. As attack vector, we used public API documentation to get prior knowledge about parameters and endpoint of URL.

3.2 Classification

API vulnerabilities were categorized on their impact and role over the situation where specific vulnerability played roles. Some of that vulnerability affects user end and some impacts its server. Like IDOR, CORS and Click Jacking impact on its user end. On the other hand Broken Authentication, Rate-Limit, Open redirect impact on server

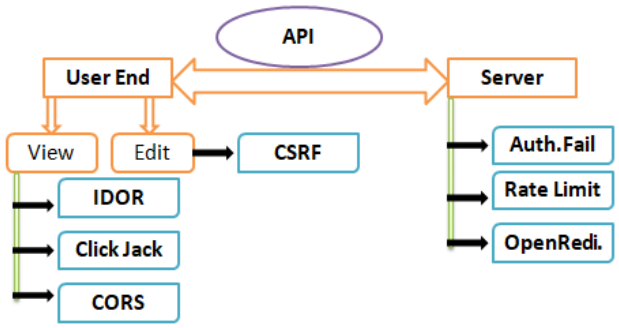

Fig. 1: API vulnerability categories depending on access area and data accessibility

```
<button type="button" onclick="load()">Exploit</button>
<p id="data"></p>
<script>
    function load(){
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function(){
            if(this.readyState == 4 && this.status == 200){
                document.getElementById('data').innerHTML = this.responseText;
            }
        };
        xhr.open("GET","http://example.com/api/userInfo/user-id",true);
        xhr.send();
    }
</script>
```

Fig. 2: A sample method of how we exploited COR

side[25] as shown in figure 1. User end divided based on their accessing level, like CSRF can modify its data but IDOR, CORS can only view confidential data.

3.3 Discovery

API flaw especially public API flaw causes various kinds of vulnerability, which can leak confidential data from an application. The IDOR vulnerability can leak data via parameter manipulation. Due to faulty API design. In the very beginning, an attacker visit the target API to guess URL endpoints that help to understand the API structure. Attacker started collecting or gather information about endpoints via footprinting and reconnaissance. We gather information about the endpoints to know what types of request it takes and what it gives as a response for that corresponding request. In this way attackers can understand the next step, whether it take permissions to create/edit/delete any data. If it permission try to escalate privileges by create/edit/delete any data as an unauthorized user.

In Figure 3 it is shown that how we exploit our APIs'. In general, we go for test IDOR vulnerability to get unauthorized access data. In that case when we got IDOR vulnerability in an API then we will try to view same data from different origin and different header. If it is possible then we will try to use those data from

different host or server. Even though CORS is not possible we can also try to exploit ClickJacking. We have found out that if one vulnerability is present then there arises a possibility to exist some other vulnerability as shown in figure 4. In this diagram, we demonstrate the probability of the attack that is related with one another.

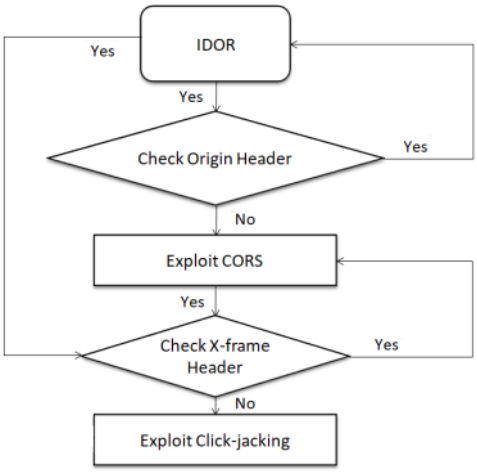

Fig. 3: How we tested to find out IDOR, CORS, X-Frame, and Click Jacking is present or not.

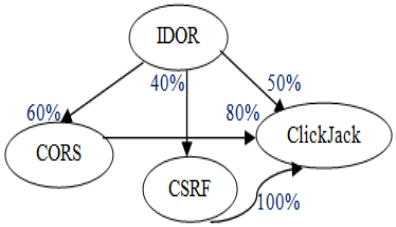

Fig. 4: Percentage to get one vulnerability if another is found.

3.4 Attack

i) Auto*:*

Selected API and targeted link set into our tools then run payload and tools. For CORS we use different tools. Every vulnerable end point returns with expected results. After exploiting these we find out their faults and the security system that API owner used.

ii) Manual:

Manually we tested those API's one by one corresponding to vulnerability. We collect their end point then we tried to determine their request with response.
Example: www.example.com/api/userInfo/{user-id}
This is the endpoint to view an user's profile. The value of **userInfo** parameter define user ID. If the user id is 50 the endpoint will be etc. www.example.com/api/userInfo/50. In the meantime if the API can't authenticate a user properly, we can change user id 50 to 51 we got another user's information.

3.5 Vulnerability and Current Security System Analysis

Various types of vulnerability were found in our sample API, we categorized them and in some cases an API has multiple vulnerabilities. Some vulnerability came out from other vulnerability. API has lots of functionality and for that reason it's made complexity with each other to fulfill their needs. API owner made their URL in a way where if an attacker was just trying to change endpoint or parameters then an attacker has a possibility of getting an output. We categorized that a certain percentage of API is not vulnerable at this moment which depends on our gathered information, and a certain percentage is vulnerable. After that those are vulnerable in specific vulnerabilities, we took them for more analysis. Then we find out and separate them based on which API is vulnerable in which vulnerabilities. Then we saw that in maximum API more than one vulnerabilities exist. Maximum API builder skip access authentication or permission, access token verification. Some of those failed server-side authentication as well. Suppose on a certain website, there exist administrator, editor, and user. Where an administrator can create/edit/delete any data, Editor can edit and delete any data, and finally user can only view his own data. But in here user escalate privileges, now not only user can view his/her own data but also user can view other users data. Also, user can delete other user's data by escalating user privileges and now the user can act like an editor.

4. Result and Discussion

Using API is easily manageable, quicker and more than productive. That is why the uses of API are increasing day to day. In the mean time those API's are getting vulnerable to many kinds of vulnerabilities. For Analysis we have tested 60 API's randomly. Where we found 53 API's are vulnerable in critical to low category vulnerabilities. And in the rest of the API's we don't find any vulnerability. Our collected API samples are from various types, which include online software related to animals, books, businesses, food & drink, health, music, university, weather and so on. These software user or admin stores various information including personal or confidential information.

In figure 6, the result is shown on our experiment. In the set of our sample API we found IDOR vulnerability in 24% API which can give provide access to confidential data. We found that 17% API are vulnerable to Rate limit is server side vulnerability. Moreover 15% API's are vulnerable to Click jacking, 10% API's are vulnerable to CSRF, 9% API's are vulnerable to Broken authentication, 7% API's are vulnerable to CORS etc. In addition to this 4% Email Spoofing, 1% Open redirect and 1% XSS vulnerability is found.

The level of the access to user's data depends on vulnerability level or category [14]. Whole dataset is divided into three categories which are:

I. Critical or highly Risk
II. Medium risk and
III. Low risk

By exploiting high category vulnerability it's possible to access a user's personal information including the email, phone number, address, location, username and etc. Also it is possible to edit that information by exploiting critical vulnerability. In medium category vulnerability an unauthorized user can login to a user account without valid credentials by exploit authentication flaws where server take an invalid request as legitimate on valid request also in here the application don't check whether the user is authorized or not. Finally in low category vulnerability an unauthorized user can pretend to be an authorized user by spoofing valid user credentials. In the given figure 7, 58%API is

vulnerable to critical vulnerability, 29% is vulnerable to medium level vulnerability and rest of the 13% API is vulnerable to low level vulnerability.

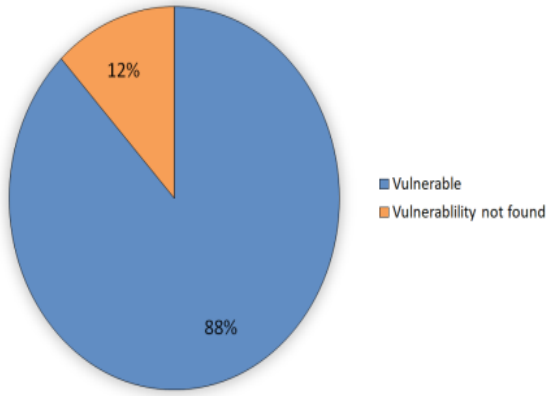

Fig. 5: Percentage of vulnerable and non vulnerable API's from sample size 60. We get 88% vulnerable and 12% non vulnerable API.

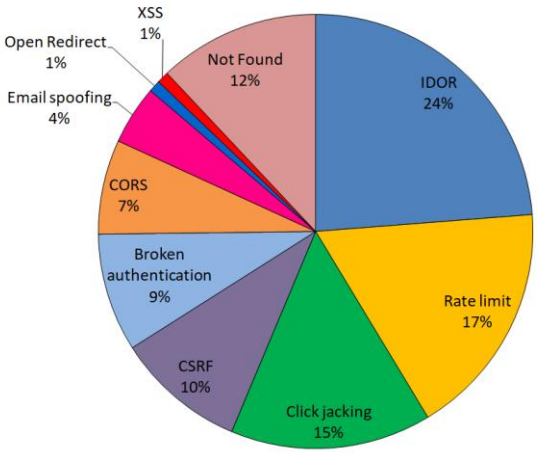

Fig. 6: Percentage of founded vulnerability on sample size. 24% IDOR vulnerability, 17% Rate Limit, 15% Click Jacking and so on, found.

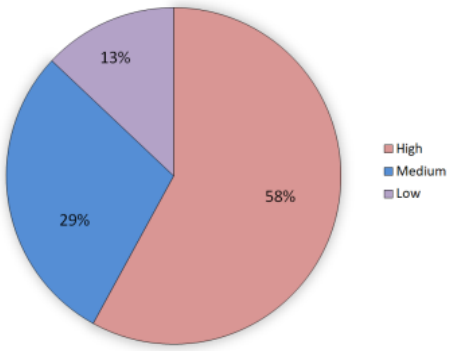

Fig. 7: Percentage of API according to highly High risk, medium risk and low risk.

Including the above 58% highly risk web applications, we have find out that inventory management websites uses the most vulnerable APIs' as shown in the following figure 8.

Sometimes one vulnerability can cause other vulnerabilities. Moreover in the figure 4 we have already shown our finding of vulnerability dependencies. If CORS attack can happen then Click jacking is going to attack easily. We want to demonstrate here if IDOR possible then 60% probability for the occurrence of CORS attack and 80% to Click Jacking. Moreover CORS can also increase probability 80% for the occurrence of Click Jacking. In the table I, this is shown. We found that:

high = animal, book, business, environment, Transportation, social
medium = animal, book, finance, health, Photography, shopping, University
low = animal, health, Photography, shopping, sports

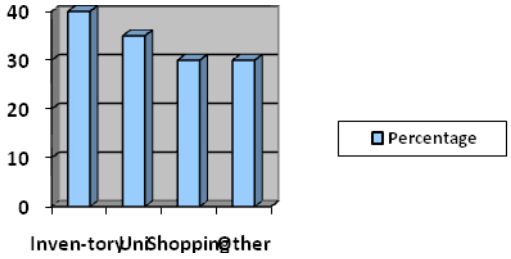

Fig. 8: In percentages which types of websites normally uses vulnerable APIs.

Table I: Shows if one vulnerability is present then percentage of presenting another vulnerability.

If Present following Vulnerabilities	Possibility to get the following
IDOR	60% Possibility to get CORS
IDOR	50% Possibility to get Click Jack
CORS	80% Possibility to get Click Jack

5. Limitation and Future Work

Now we made our study on more than 60 samples. To achieve more perfect result we need to consider more samples. Moreover we considered public APIs only as we need permission to test other APIs.

Again we discussed only the current status but we need to solve these problems. In future, we will propose a new security layer to easily prevent critical vulnerability more effectively.

6. Conclusion

In this paper, we determine the current status of vulnerabilities present in public APIs. Some of the APIs have basic protection like input validation. We identified their present provided security status. Around 88% APIs are affected by many types of vulnerabilities. The number of vulnerable APIs can be increased. If that vulnerability rate gets out of control there is an enormous possibility of data breach. To increase usability of API, we have to provide its proper security otherwise its purpose will not be fulfilled. To ensure APIs security we have to work together in production level and user level.

Acknowledgement

This is a self funded project. There is no conflict of interest.

References

[1] Kim S.S., Lee D. E., Hong C. S., "Vulnerability Detection Machanism Based on open API for Multi User's Convenience" Kyung Hee University.

[2] Myer's B. A., Stylos J., "Improving API usability" Human-Computer Interaction Institute School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891.
[3] Deng Z., Saltaformaggio B. , Zhang X., Xu D., "iRiS: Vetting Private API Abuse in iOS Applications", Department of Computer Science and CERIAS Purdue University, West Lafayette, IN 47907.
[4] Alqahtani S. S., Eghan E. E., Rilling J.," Recovering Semantic Traceability Links between APIs and Security Vulnerabilities: An Ontological Modeling Approach", Concordia University Montreal, Canada.
[5] Thomas D. R., Beresford A. R., Coudray T. , Sutclie T. , and Taylor A., "The Lifetime of Android API vulnerabilities: case study on the JavaScript-to-Java interface," Bromium, Cambridge, United Kingdom.
[6] Mao Y., Chen H., Zhou† D., Wang X., Zeldovich N., and Kaashoek M. F.," Software fault isolation with API integrity and multi-principal modules", MIT CSAIL, †Tsinghua University IIIS.
[7] (5thJune 2017) Top 5 Vulnerabilities In APIs. [Online]. Available: https://datafloq.com/read/top-5-vulnerabilities-in-apis/2876.
[8] (10thAugust 2017) Viber API Documentation. [Online]. Available: https://developers.viber.com/docs/api/rest-bot-api.
[9] (12th August 2017) Web API tester [Online]. Available: http://stoplight.io/platform/scenarios/.
[10] Zhang M., Duan Y., Yin H., Zhao Z.," Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs", Syracuse University, Syracuse, NY, USA.
[11] (8th july 2017) Documentation and Test Consoles for Over 500 Public APIs [Online]. Available: https://any-api.com.
[12] (19th july 2017) Open API [Online]. Available: https://www.getpostman.com/docs/postman_for_publishers/public_api_docs.
[13] (20th june 2017) Recent news about security [Online]. Available: https://thehackernews.com.
[14] Sami A., Yadegari B., Rahimi H., Peiravian N., Hashemi S.," Malware detection based on mining API calls", Ali Hamze Shiraz University, Shiraz, Iran
[15] (20th june 2017) REST Security Cheat Sheet. [Online]. Available:https://www.owasp.org/index.php/REST_Security_Cheat_Sheet
[16] (1th may 2017) Bug bounty platform [Online]. Available: www.hackerone.com
[17] (10th may 2017) List of public API. [Online]. Available: https://github.com/toddmotto/public-apis
[18] Johari R., Sharma P., "A Survey On Web Application Vulnerabilities(SQLIA,XSS)Exploitation and Security Engine for SQL Injection", 12 International Conference on Communication Systems and Network Technologies
[19] Bhuiya T., Alam D., Farah T., Evaluating the Readiness of Cyber Resilient Bangladesh, January 2016, International Journal of Internet Technology and Secured Transactions 4(1)
[20] Rexha B., Halili A., Rrmoku K. and Imeraj D., "Impact of secure programming on web application vulnerabilities," 2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS), Bhubaneswar, 2015, pp. 61-66.
[21] Bhuiyan T., Alam D., and Farah T. (2016). Evaluating the Readiness of Cyber Resilient Bangladesh.Journal of Internet Technology and Secured Transactions (JITST), Vol. 4, No. 1, ISSN 2046-3723.
[22] Begum A., Hassan M. M., Sharif M. H., Bhuiyan T., "A study on RFI and SQLi based on Local File Inclusion Vulnerabilities in the Web Applications of Bangladesh" , International Workshop on Computational Intelligence, 12-13 December-2016
[23] Moussaid N.E.E. and Toumanari A., "Web Application Attacks Detection: A Survey and Classification", "International Journal of Computer Applications (0975 – 8887) Volume 103 – No.12, October 2014"
[24] Ami P. V. and Malav S.C., "Top Five Dangerous Security Risks Over Web Application", "International Jurnal Of Emerging Trends & Technology In Computer Science,2013 "
[25] Chakraborty R., Datta A., Mandal J.K., "Secure Encryption Technique (SET): A Private Key Crypto System", "International Journal of Multidisciplinary in Cryptology and Information Security", Volume 4, No.1, January – February 2015
[26] Kumari M. S., Shrivastava D. M., "A Study on the Security and Routing Protocols for Ad-Hoc network", "International Journal of Advanced Trends in Computer Science and Engineering", Volume 1, No.3, July – August 2012