International Journal of Security and Its Applications
Vol. 10, No. 9 (2016), pp.55-64
http://dx.doi.org/10.14257/ijsia.2016.10.9.07

NoSQL Injection: Data Security on Web Vulnerability

Hemn B. Abdalla[1], Guoquang Li[1], Jinzhao Lin[1] and Mustafa A. Alazeez[2]

[1]*School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, China, Chongqing*
[2]*School of Computer Science, Huazhong University of Science and Technology, China, Wuhan*
[1]*hemn.db85@yahoo.com*

Abstract

Web sites can be dynamic and static sometimes a combination of both. To assure security all the websites need protections to their database. This paper proposes a static analysis of various technique of detecting recently discovered web-based vulnerabilities such as cross-site scripting and HTTP splitting attacks with NoSQL injection. Today, the situation is¬ better, and traditional databases have introduced built-in protection mechanisms. But does this mean that NoSQL systems are immune to injections? Our study shows that although the security of the query language has largely improved; there are still techniques for injecting malicious queries. Some works already provide reports of NoSQL injection techniques. We also show the methods of accessing the dark web. We use TOR browser for the explanation of accessing dark web. The user entering into the website with different format rather than the original will be noticed, and they are not allowed to access our website. Instead of passing SQL queries into the input fields the user can also directly pass the objects inside it. Once the user enters in, with the actual format and attempts to upload .exe file the website tends to block that user's account.

Keywords: *NoSQL, TOR Browser, Dark Web, Vulnerability*

1. Introduction

The vulnerability is nothing but a weakness that allows an attacker to reduce a system information assurance. In earlier versions of MongoDB, there will be a variable name 'db' in the context of the JavaScript that runs in $where variable [1]. The db variable will perform Boolean injection through which the extraction from the database can be done. So it is clear that JavaScript can allow injection in some places. The attackers try to access the sensitive information from a database of the web page by first generating a NoSQL query that will cause the database parsing function to fail and followed by applying those same queries to the desired database. Rather than the SQL injection there another method that uses other formats which are a NoSQL injection. A static break-in strategy is to try to access information from a database by first generating a query that will cause the database parser to fail. Such an approach to gaining access to private information is called SQL injection. The Dark Web (also known as Darknet) is a subset of the Deep Web that is not only indexed, but that also requires something special to be able to access it, *e.g.*, specific proxy software or authentication to gain access [2] [3]. The Dark Web can be reached through decentralized nodes on some networks including Tor (short for The Onion Router) or I2P (Invisible Internet Project). Tor, which was initially released as The Onion Routing project in 2002, was created by the U.S. Naval Research. Tor's users connect to websites "through a series of virtual tunnels rather than making a direct connection. Tor protects the user against a common form of Internet surveillance. It is mainly used to infer

ISSN: 1738-9976 IJSIA

who is talking to whom over a public network. Knowing the destination and source of your Internet traffic allows others to track your interests and behaviors.

This project mainly involves three major modules. First shows the main concept of NoSQL injection which is one of the most critical web vulnerabilities. How NoSQL and cross-site injection happens? It clears the idea of this. The last one is all about the TOR browser. What happens if a person enters into our webmail through TOR browser and another format of web address? Next one is the blocking of mail while uploading the .exe file. Here it checks the file format and blocks the account if he/she is an uploading .exe file.

2. Methodology

The main process explained here will be checking the website format. The format includes some minor changes. That will be captured and discriminated accordingly. This type of discrimination is also called as clustering or grouping accordingly. By this way the attacker can be differentiated among the other users and process is done. This process is expressed here with the web page clustering method. The main aim of our project is to identify the NoSQL injection that allows the user to the attacker to bypass authentication.

We can generate the request that corresponds to the NoSQL injection that leads to loading error. Here we are creating 3 set of requests; they are R_r, R_s, and R_t. We need to check whether the response is a rejection page or response page. For the above requests S_r, S_s, and S_t are the responses.

The principle of our algorithm is then as, R_t requests whose responses are not similar to any of the replies from S_r, S_s are considered as valid NoSQL injection. To access the similarity between the pages returned by different requests, we need to use some other classification technique based on distance presented in next subsection. The normalize difference method is used to find the difference between two pages. Let a and b be the two responses of length m and n. Let i^{th} character in 'a' be a_i and Let j^{th} character in 'b' be b_j.

$$d(a,b) = \frac{diff(a_i, b_i)}{(n+m)} \quad (1)$$

$$diff(a_i, b_i) = n - i + m - j, \text{ if } i = n \text{ or } j = m \quad (2)$$

$$diff(a_i, b_j) = diff(a_{i+1}, b_{j+1}), \quad \text{if } a_i = b_j, i < n, j < m \quad (3)$$

$$diff(a_i, b_j) = 1 + min(diff(a_{i+1}, b_j), diff(a_i, b_{j+1})), \quad \text{if } a_i \neq b_j, i < n, j < m \quad (4)$$

Algorithm:

Step 1: Generate Request

Generate request makes the request that corresponds to the NoSQL injection that leads to loading error.

R_r → Request generated from words {[a-z A-Z 0-9] +}

R_s → NoSQL injection requests inappropriate for the injected point.

R_r → NoSQL request that is constructed to generate Execution pages.

Step 2: Responses:

S_r, S_s and S_t → Responses associated with R_r, R_s, and R_t.

Step 3: Finding the similarity between the pages:

To access the similarity between the Response pages, we need to locate the distance for response pages.

a,b → response of length m, n.

ai → ith character of a.

bj → jth character of b.

$$Distance, \ d(a,b) = \frac{\text{diff}\left(a_{1,\text{till}}\, b_{1}\right)}{(n+m)}$$

3. Implementation Modules

Implementation is one of the critical stages of the project; it is nothing but a change of working system from the theoretical system design. The efficient performance of the scheme entirely depends on this stage. Hence, it is said to be the most critical juncture of the project development. The usage of NoSQL database store not only eliminates the SQL language entirely and improves the security but also simplifies the development. It relays on simpler and structures query mechanism in the face of JavaScript. The implementation phase involves investigation of the existing system, careful planning and its son constraint implementation, the design of methods to achieve changeover and evaluation of transition methods. To query the login details MongoDB can use the following method:

db.users.find ({username: username, password: password});

3.1. Main Architecture of the Project

This is the main architecture of the program. The main process explained here will be checking the website format. Depending upon the website format the admin will do the remaining process. The format refers in “www” & “ww1”. The user who enters through “ww1” will be fixed as an attacker. Both the formats will be allowed to access the database initially [4] [5]. After that, if the user was entered through “ww1” then the information will be sent to the admin. The information of the attacker will be captured, and the admin will block it. After allowing the user through “www”, he\she can access database and also can do any operations. Once the user was trying to upload a .exe file, it will be blocked by the admin. Figure 1 clearly shows the above-given process.

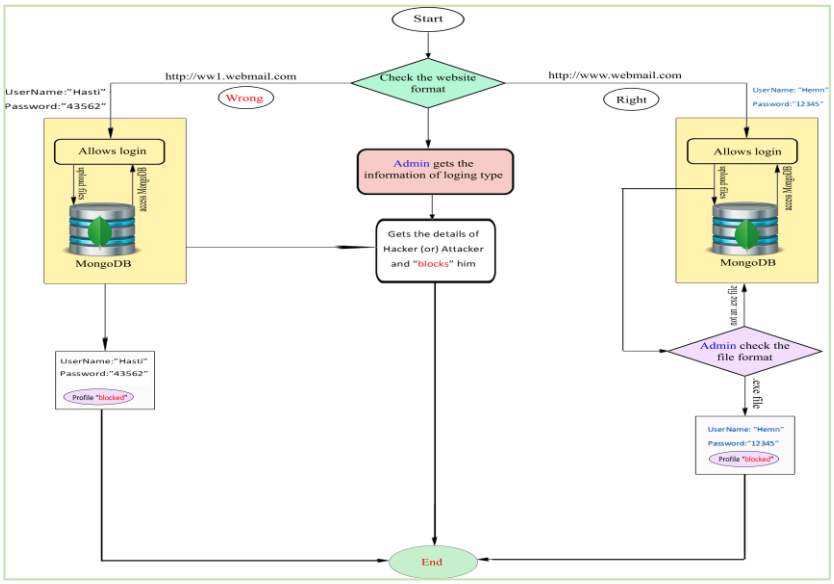

Figure 1. Main Architecture

3.2. NoSQL Injection Execution

Here two different types of Injection attacks are explained to make the topic clear:

Injection attacks are nothing but the result of a Web application sending malicious data to the server. The attackers try to access the sensitive information from a database of the internet page by first generating a NoSQL query that will cause the database parsing function to fail and followed by applying those same queries to the desired database [6].

Cross-Site Scripting, or XSS, is the most prevalent security flaw that Web applications are vulnerable to. In an XSS attack, a malicious code in inserted into the database [7]. When the injected malicious code is executed in a visitor's browser, it can manage the browser to do whatever the attacker needed. These types of attacks include hijacking the user's session, or redirecting a user to another site or installing malware.

3.3. Object Injection

Object injection is a kind of vulnerability that can allow the attacker to do different kinds of malicious attack. Instead of passing SQL queries into the input fields the user can directly pass the objects inside it. The user can pass an object as a value into a vulnerable call, resulting in object injection inside the application scope.

3.4. Dark Web (Tor Browser Attack)

Dark web is nothing but a subset of Deep web. The deep web is called as the subset of the internet which is not indexed by the most of the search engines. The Dark Web can be reached through decentralized nodes on some networks including Tor (short for The Onion Router) or I2P (Invisible Internet Project). Tor protects the user against a common form of Internet surveillance. It is mainly used to infer who is talking to whom over a public network. Knowing the destination and source of your Internet traffic allows others to track your interests and behaviors [8].

The primary process involved here was the format checking. The permission to the user will be based on the website format through which the user enters inside. "www" &"ww1" are the example formats we have taken for illustration. The person who comes through"ww1" will be fixed as an attacker. Both the formats will be allowed to access the database initially. If the user enters through the second format, the admin will be notified, and he will block all these account details. The information of the attacker will be captured, and the admin will block it. Figure 2 clearly shows the above-given process.

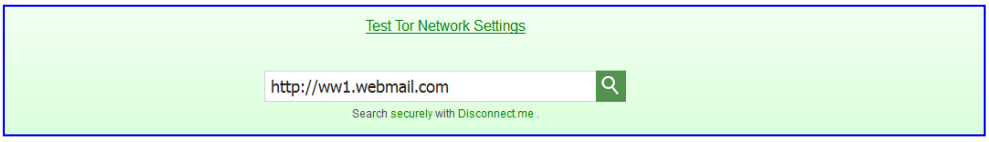

Figure 2. Tor Browser Attack Module

3.5. Cross-site Injection Attack

Cross-site scripting nothing but a code injection attack on the client-side where the attacker will inject malicious scripts into a website or application.

- The attacker injects payload in website database by submitting a vulnerable form with some malicious JavaScript.
- The victim browser page loads with the payload that has been injected as a part of the HTML body.
- The script will send the victims cookie to the attacker's server. As shown in figure 3.

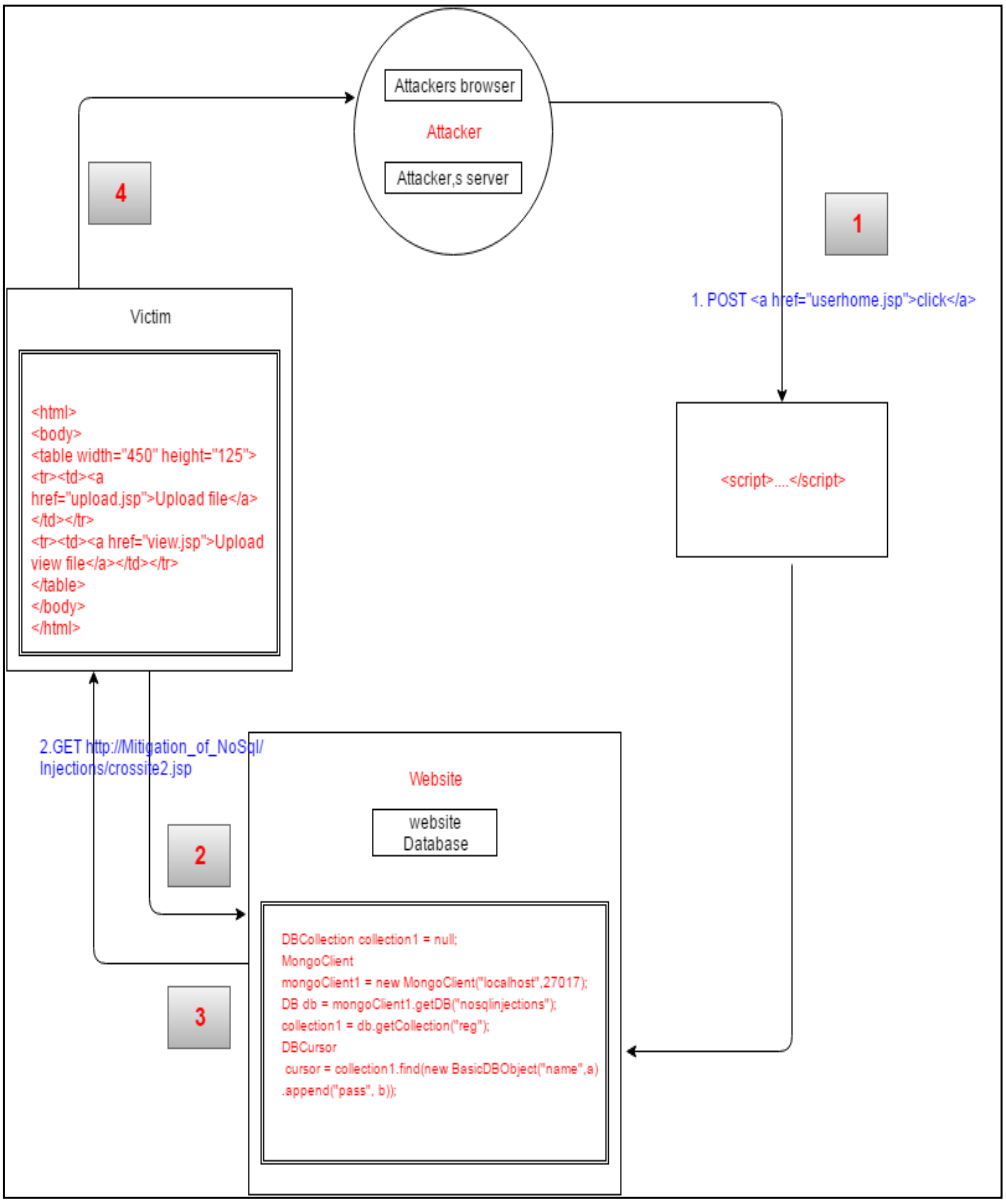

Figure 3. Cross Site Injection Diagrammatic Representation

- The attacker will now extract the victim's cookie when the request arrives at the server.
- Finally, the user can use the victim's stolen cookie.

3.6. EXE File Uploads Attack

Once the above processes are completed, the user will be allowed to access the database in both the form of uploading and downloading. While uploading .exefile, the user's account will be blocked automatically. This is one part of avoiding the web vulnerabilities. As shown in Figure 4, 5.

Figure 4. Uploading .exe File

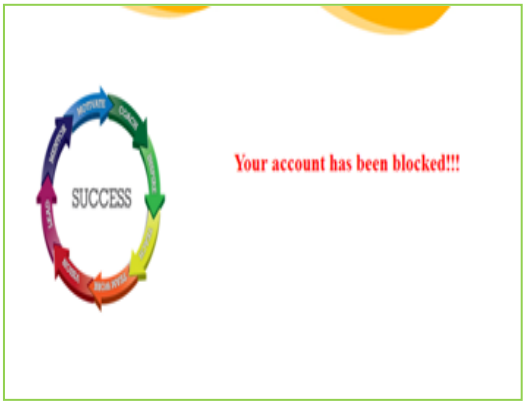

Figure 5. Account Blocked!

The above-given samples of web Vulnerability with NoSQL have been subjected to analysis. In order to decrease the web vulnerabilities issue, once need to take steps to the above-mentioned solution. It gives us another option to get not involved in vulnerabilities issue.

With the rising adoption of server-side scripting, one can expect JS injection vulnerabilities caused by an invalidated user to become prevalent and the exploiting techniques. All the above modules that easily show the exploitability of the vulnerability. As shown in Figure 6.

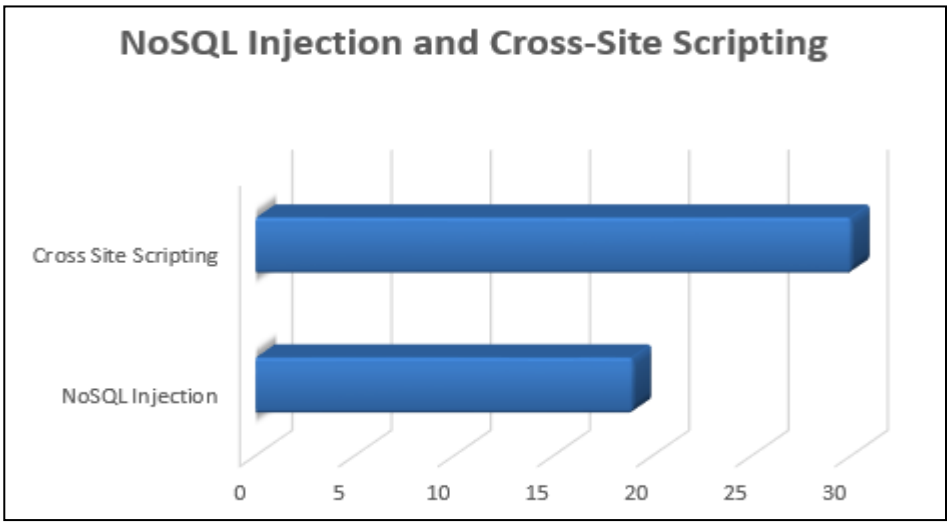

Figure 6. The Percentage Difference Between The NoSQL Injection and Cross-Site Scripting

3.7. Code Implementation

(1) Dark Web (Tor browser attack):

```
Start
stringlen <- lenghofstring
        i Patlen
top;
if  i > Stringlen then return false
         j <- Patlen
loop;
```

```
if string(i)=path(j) then
        j<-j-1;
        i <-i-1;
goto loop,
close;
      i <- i max (deltai (string(i)),delta2(j))
goto top.
End
```

(2) EXE File uploads attack:

```
Start
String input file=".exe";
Sting  Blocked="input file", Proceeded="Success";
If(.exe=input file)
{
      Pint(Blocked);
}
else
{
     Print(Proceeded);
}
End
```

(3) MongoDB (NoSQL Database):

MongoDB, are quickly grown to become a popular database for web applications and Mobile application, NoSQL is fast data access. Is an entirely appropriate for a Node.JS applications let us write a JavaScript to any client, back-end, and the database layer.

MongoDB is being used in many critical projects and products; besides, the inserted support for location queries is a bonus that's hard to be dismissed, faster to enter and retrieve data's. Its schema-less kind is a more serious mate to our permanently evolving data structures in web applications and mobile application, relatively new to the database market. Our Data is Location Based; MongoDB has special built-in functions, so finding related data from particular locations is fast and accurate.

Shows our accounts blocked in MongoDB :

```
{
   "_id" : ObjectId("558147b98798988b3025685c"),
   "UserName" : "Hemn",
   "Password" : "12345",
   "email" : "hemn.db85@yahoo.com",
   "Mobile" : "1987265333",
   "dob" : "01-01-1985",
   "Address" : "CQUPT",
   "Status" : "blocked",
   "url" : "http\\:ww1.webmail.com",

}
```

4. Future Scope

Nowadays web application becomes more popular and important in our day-to-day life. At the same time vulnerabilities that endangered the personal data of users are regularly discovered. Various methods can be available to create security for web-based vulnerabilities. To save money and time we are encouraged to use automated tools for vulnerability assessment.we have illustrated some of the preventive methods for the prevention of cross-site scripting and SQL vulnerabilities in web applications. Our analysis was good and by following these methods, we can defend web applications from this modern web threats. The concept of a secure internal network has been revoked in countless attacks on enterprises such as the Adobe password breach, RSA Security, and Sony. The internal network is bound to be infiltrated at some point, and it's our duty to make it as difficult as possible for attackers to gain advantages from that point on. This applies especially to some NoSQL databases that are relatively new and lack role-based permissions, which means anyone can execute anything on them (as was the case for Me cached). For this, a strict network configuration is recommended to ensure that the database is accessible only to relevant hosts, such as the application server.

5. Conclusion

Earlier, we saw best illustrations about how the traditional network security solutions are not effectively worked against the common vulnerabilities that present within a Web application framework.However, because these tools explained above do not protect against Web application vulnerabilities,it doesn't mean that there is no defense against these threats. On the contrary, the Firewall solution of a web application provides protection that meets compliance regulations set by one of the most stringent industry security standard (Payment Card Industry Data Security Standard).

Mitigating security risks in NoSQL deployments is important in light of the attack vectors we present in this article. Unfortunately, code analysis of the application layer alone is insufficient to ensure that all threats are mitigated. Three trends make this problem even more challenging than before. First, the emerging cloud and big data systems typically execute multiple complex applications that use heterogeneous open source tools and platforms. These are commonly developed by open source communities and, in most cases, don't undergo comprehensive security testing. Another challenge is the speed of modern code development with DevOps methodologies, which aim to shorten the time between development and production. Finally, most application security testing tools can't keep up with the fast phase with which new programming languages are adopted.

Acknowledgment

This research work is supported by University Innovation Team Construction Plan of Chongqing, the National Science Foundation of China under Grant No. 61301124, 61671091.

References

[1] P. Noiumkar and T. Chomsiri, "A Comparison the Level of Security on Top 5 Open Source NoSQL Databases".
[2] A. Ron, A. Shulman-Peleg and E. Bronshtein, "No SQL, No Injection? Examining NoSQL Security", ArXiv Prepr. ArXiv150604082, **(2015)**.
[3] M. Malik and T. Patel, "Database Security - Attacks and Control Methods", International Journal of Inf. Sci. Tech., vol. 6, no. 1/2, **(2016)** March, pp. 175-183.
[4] A. Petukhov and D. Kozlov, "Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing", Comput. Syst. Lab Dep. Comput. Sci. Mosc. State Univ., **(2008)**.

[5] N. Singh, A. Jangra, U. Lakhina, and R. Sharma, "SQL Injection Attack Detection & Prevention over Cloud Services", International Journal of Comput. Sci. Inf. Secur., vol. 14, no. 4, pp. 256, 2016.
[6] M. Mahdi and A. H. Mohammad, "Using Hash Algorithm to Detect SQL Injection Vulnerability", **(2016)**.
[7] M. Factor, D. Hadas, A. Hamama, N. Har'El, E. K. Kolodner, A. Kurmus, A. Shulman-Peleg and A. Sorniotti, "Secure logical isolation for multi-tenancy in cloud storage", Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium, **(2013)**, pp. 1-5.
[8] S. Mattoo and R. Kumar, "Server Side Attacks On Web Applications", **(2016)**.

Authors

Hemn Barzan Abdalla received the master degree in computer science from Huazhong University of Science and Technology, in 2014. Currently, he is a Ph.D. student at Chongqing University of Posts and Telecommunication (CQUPT). His research interests include Big Data and Data security, NoSQL, Application.

Guoquan Li received his Ph.D. degree in College of Communication Engineering from Chongqing University (CQU), Chongqing, China in 2012. He is currently a lecturer at Chongqing University of Posts and Telecommunications, Chongqing (CQUPT), China. His research interests include digital baseband signal processing, multi-user MIMO system, coding theory and technology.

Jinzhao Lin received his Ph.D. degree in 2001 from Chongqing University. Since 2003, he has been a professor at Chongqing University of Posts and Telecommunications (CQUPT). His current interests include wireless communications and ASIC design.