COLLEGE OF SAINT BENEDICT SAINT JOHNS UNIVERSITY

SENIOR RESEARCH SEMINAR

The Anatomy Of SQL Injections

Author:
Zachary EICH

Professor:
Dr. Mike HEROUX

December 15, 2014

Abstract

SQL Injection is an increasingly common method hackers use to gain access databases and to your private data, and using it against you. Database defenses are still being hacked even after over a decade of major security breaches attributed to SQL injections alone, and costing companies and their beneficiaries. The amount of injections in hacking is increasing and the methods of delivery are changing to make them harder to detect. Security providers are using code level defense like parameterization, and concatenated statements, and are proving effective at detecting injections before their execution. Parameterized statements are the best defense against SQL injections and while effective are not widely used today. Awareness needs to be spread in the Security industry preventing the use of basic SQL injection.

CONTENTS

LIST OF FIGURES

I. Introduction

Hackers have been around for the past 50 years, making financial gains for themselves at the cost of others. In the 1960s hacking was a simple as making free long distance phone calls.[21] Not until late 1980 was the Computer Fraud and Abuse Act passed so that government Officials had a jurisdiction in arresting those exposing unprotected glitches. SQL Injection or SQLI is the use of dynamic Strings, to manipulate the Query sent to a database in the Structured Query Language. It has been uncovered in the last few years as the prime method for hacking on the internet, targeting the database behind websites.[5] These databases are profitable to anyone who holds the information within. Facebook alone asks for addresses, phone numbers, emails, contacts, images, and even credit Card information if given.[23] This information is a virtual projection of ourselves and can be used against us to create false credit cards and take out false forms financial spending.

SQLI it is often used for hacking because it is simple enough for the younger generations to understand, and yet it can combined with other hacking methods to become an advanced form of hacking. Teenagers have been able to use it to rip open databases, one example is Jeremiah Jacks. He effectively broke into guess.com and accessed hundreds of users credit cards. Instead of exploiting them however, he reported the issue and is working as a Security consultant now.[19] SQLI is still effective because of the changing ways it is presented into databases.[5] Here we will see that with the advent of new Injection methods the use of SQLI will increase.

II. Survey SQLI

A. Status of SQLI

Nataan describes SQL servers as functionally rich, and inherently insecure [17]. Databases were once considered secure in a locked room, only accessible through authentication (username and password), and blocked by a firewall. But SQL injection can pass right through all these protective measures directing itself to the back-end of a database to execute freely. These measures were once effective at blocking all environmental and illegitimate user hacks, keeping accessibility to the database at a bare minimum. This restriction to the servers only granted access to the employees and those utilizing the database. Now the majority of business is conducted online through the internet, giving almost anyone access to databases. Access can be tracked through audits on users (admin or not) and the management systems. Audits contain information on user logs and grants, helping to track an injection hacker. User logs will tell you when a user has accessed a database, and what information they have handled. Grants are useful to track the change of permissions, and for most users it does not change often. In the case of injection, a hacker would either access the database as an admin or grant a user abnormal privileges[11]. Audits catch some injections, but SQL injection can still pass and access the database if the server is not set up properly. The injection can bypass all of these audits by acting as the database itself, given it has the right permissions.

B. Introduction Phase

SQLI was not as popular when it started. The use of databases was not as heavily relied on as it is today. Modern websites depend on database to protect the larger amount of user information stored in them.The implementation for protection can be done while creating a database and updates can patch old databases to keep injections out [7]. While it had been an exposed issue, not may had hackers caught on to using it as an exploit. [21]

C. Catalyst for Development

More people are shopping online every day, increasing the number of users and data stored in a database.[2] With all this data coming into a database Auditors will need to spend more time sifting through databases to find unwanted users i.e. hackers. Common sense will dictate limiting the users ability to query the database, without limiting access to the consumer. Auditors will also need to maintain a very small group with admin privileges, keeping the span of control limited and manageable. Those stored as a administrator should also be stored in a separate database as the normal user, to decrease the access to an administrator database.

OWASP is an Open Source Web Application Security, that creates an environment where programmers can go to discuss the best defense against hackers who use methods such as Injection[20]. While this also gives hackers access to the methods against injection it still is effective at spreading awareness and defense techniques. is similar to the Def Con conferences, held to educate the computer science community that shows interest in the hacking industry.

III. Ethical Considerations

A. Ethics in SQLI

The three categories considered in modern hacking are Activists (dubbed Hacktivists), Cyber Criminals, and Cyber spies. While the issue of hacking immediately implies a sense of distrust and wrongdoing, the benefits to studying hacking are a greater understanding of modern computing and better defense against the morally inept. This is the focus of the first group the Hacktivist. Aligning themselves with the defense and the color blue, they are what represent the good side of the hacking realm.(sans) The red team is the opposite, often known as the offensive team, set on obtaining information or crashing the others Operating System. While both teams compete the end result is a better understanding of computational defense. Conferences like Def Con provide similar learning environments and forums for distribution of these events discoveries. They even go as far as creating simulators for high school students to hack into, furthering interest, and increasing awareness in the next generation. Cyber Criminals are the biggest issue in ethical hacking. Their motivation lies strictly in a net gain whether it be finical or otherwise. Their use of SQL Injection is strictly for access to the data, manipulating and stealing it for personal gains. These are the people Hacktivists train to beat. Cyber Espionage is focused highly in our world today. China's Axiom group has recently been discovered to be behind many government agency attacks and private companies.[16]

IV. Technical Analysis

Looking at SQLI from a technical analysis we can see that many forms exist and can be used to manipulate databases. Basic front end injection and Blind SQLI are common methods still used today using the text boxes on a web page. These methods have their own unique characteristics but still involve the mishandled dynamic query sent to a database.

A. Front End Injection

The page displayed for viewers is often referred to as the front end of a website. The back-end, is where the computer interacts with servers. No matter what a individual does online, their actions will query a database. Whether it is clicking a link or typing in a search bar, the web page creates the query for the local server. When browsing the web queries sent from your computer to not only to your local server but to the server where your intended website is stored on.[1] Where SQL now can inject is in the search box you type in. The way the web is set up is to communicate with databases so we do not have to, as shown below.

Fig. 1. Demonstrates the path of communication from a mouse click to querying a database.

Figure 1. is the relationship of a users device to a Server. Code sent is queried through web interface that sends the query to the Server with the use of a SQL Query Statement. A very simple diagram, it shows how what you do interacts with the server. First off, information we see is displayed in a browser, which is a simply constructed form that has pre-established settings, looks, and options. What you select on the web interface is read into the Query form, for this example Citrix XML. The query form could be XML, MySQL, PHP, HTML, or any language that construct a websites reactions, all of which are ways that the browser can communicate with the server. Once it receives input it generates a query looking like the one shown below in Figure 2.

```
Query 1
Select * from MyUsers where user.name Like "jack"
Query 2
'Select * from MyUsers where user.name Like "jack" --
```

Fig. 2. Shows two forms of a query sent to a database. The second is a malicious form ending in a delimiter to stop compilation.

The dangerous form of this query would look more like the second query. This Query will stop the complier and ask for a user name similar to user Jack, therefore the compiler finds its own user to login as, leaving the hacker to focus on the password. In SQL – (double dash) serves to comment out the remainder of code preventing it from continuing its intended statement.[4] Understand that when you type into a search box, you can be given unlimited choice for characters you wish to type. This can trick the complier that is processing the query, and make it stop during its execution. By starting with an the first thing the complier thinks is that it has reached the end of the given String. When followed by or 0=0 / the execution is intent on adding another scenario known to be true so that it can continue without a user. Since we know 0 will always equal 0 this statement will always

be true and therefore, always execute. The remaining backslash serves to comment out the remainder of the query this code intended for database accessed with java. This hides an injection as it accesses a serve but lets look at injects that do not know how the server is defended.

B. Blind SQLI

Hackers might not know the structure of the database they are trying to breach. Blind SQLI is the method that overcomes this issue by giving bits of information on the database buy using informative errors, time delays and print statements.[9] The best example of the latter is the @select version which can display the information in Figure 3.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'Microsoft SQL Server 2000 - 8.00.534 (Intel X86)
Nov 19 2001 13:23:50 Copyright (c) 1988-2000 Microsoft Corporation
Enterprise Edition on Windows NT 5.0 (Build 2195: Service Pack 3)' to a
column of data type int.
```

Fig. 3. This helpful reply is a response from the database. THe information provided give a hacker an exact idea of what version and provider the database is operating on.

The key with blind injection is to send as many queries as possible and create as many responses similar to the ones shown. This however shows its own weakness, because if countless queries are what decipher the security, obviously limiting not only the number of queries but also the time in between queries helps to stop injection. Some errors may even return the year and the version of security or Systems in place.[1] When using Blind SQLI on a URL, you change a character at a time to see what page returns. Some Depending on the page received, the effect of the injection is displayed. pages will return nothing, often the symbol for success, and some pages will redirect back to the main page, a default safety. You might be wondering how time can have an affect, on reading a system. IN this scenario, our hacker has decided he doesn't know much about the security systems capabilities, however he knows that it logs the number of queries, and the amount of errors, defeating simple blind injection. However if we introduce a time delay we have time to not only test if the query is correct but to cover up if it is incorrect. Delays can be set in the abstract forms of code as well, causing a complier to print a large amount of times, pause execution, and create infinite loops. They can not only inform you of the database but provide a Denial of Service while the complier is stuck in a loop. [12]

C. SQLI Prevention

To better understand SQL Injection, some programmers have turned developing SQLI creating software to test database security. At team headed by Adam Kiezun developed a program called Ardilla, using the existing code a developer would like to publish and running it through all possible values of SQLI known.[13] By testing the rereleased code developers are catching the SQLI statements that are still evading the basic code. If the developers programmed their defenses incorrectly, Adrilla can find the holes. It can track the flow of data to prevent Cross Site Scripting (the use of hacking through data packets) as well as SQLI. Ardilla can be helpful in the detection of Content Management Systems(CMS) insecurities. Drupal a well-known CMS was hacked recently OCT 14 using simple SQLI, to crack the adding of unauthorized users. These hacks used ; to stop compilation and than proceeded with methods using the same code displayed earlier to not find users but add and remove users. Users were than created with different levels of control depending on their purpose i.e., flood the system or steal it. One way Drupal can circumvent Blind injection is what they offer in their modules called Clean URL.

Fig. 4. Drupal 7 uses clean url's to prevent minor injections and hide the remaining page information. This helps hide database structure from intelligent hackers.

The following shows the effect on URLs displayed by their websites in order the prevent access from changing Characters in the URL. (drupal)

V. Future Trends

A. Trends in Technique

In research, SQL Injection is often paired with Cross Site Scripting (XSS). They both involve the input of malicious code into code, and result in corrupted database operations. [22] When combining the two methods XSS and SQLI you project the Injection within a packet that is executed when an Admin logs in. This creates another bypass to not only firewalls but most other detection methods, simply because the injection hiding in safe packets. Other combinations can include time delays and automation showing how well thought out new Injections are getting.

In the Technical Analysis we discussed the expanding list of code that is malicious and how each character down to a null value, can be harmful. This goes to show that no character accepted into a query should be overlooked. We can see that as you add a defense against a style of malicious coding, they begin to search deeper into the meaning each character can have on compiling.

B. Where is SQL Injection headed?

Security Magazine does not believe SQLI will be an issue forever. Reporter Adam Green states the Mobile Market is a much greater target. [8] SQLI however is a threat that can affect the mobile market as well. Having just as much access to dynamic Queries the mobile market stores the same information that websites do. The threats to the web market are just as risky to the cellular market. [6]

Online forums like Hackforums.net have an increase in SQLI conversations. The online hacking community still shows interest and likes to post about future success, to play onlookers into helping them out.[15] The community was destroyed however its 250,00 users will more than likely find another website to cooperate.

C. Facts

SQL Injection is still present even after its discovery in 1999. Tech Republic reports three interesting facts.

”Sixty-five percent of organizations represented in this study experienced a SQL injection attack that successfully evaded their perimeter defenses in the last 12 months. Twenty-one percent (the largest group percentage-wise) stated it took up to six months to detect the attack, and twenty-one percent (the largest group percentage-wise) said it took a month to contain the attack.”[18]

Dark Reading also reports high density of attacks, with 38% increase of injections using old techniques.[3] They also report from Verizons Data Breach Investigation Report 80% of attacks are SQL injections. This goes to show that while SQL Injection can be caught, hackers are getting away with using the same strategies to inject on bases. What code does not work for some sites will work on others simply due to the large number of targets, and the little knowledge put into their defenses.

In accordance with OWASP, Hackmageddon also posts a clear leader in hacking techniques.[21] Reports of the number of Cyber attacks have increased by as much as 73%.[14] High Tech also boasts SQLI as 35% of the attacks Techniques. [15]

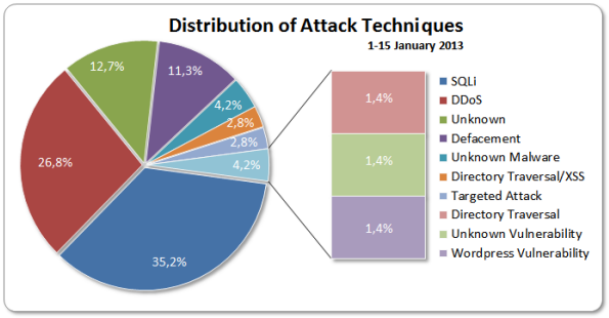

Fig. 5. High Tech’s Analysis of the most common methods used in moder computing.

D. Limitations

The role of dynamic queries (user input) is too large on the Internet therefore we need them in order for it to be useful. Defenses like firewalls and other software exist, but the best defenses are already known. However with every new defense, we wrap our databases in more code, giving surface for injections to attack.[10] Therefore the best code is the minimal it takes to block out current injection a parameterized statement.

VI. Demonstration

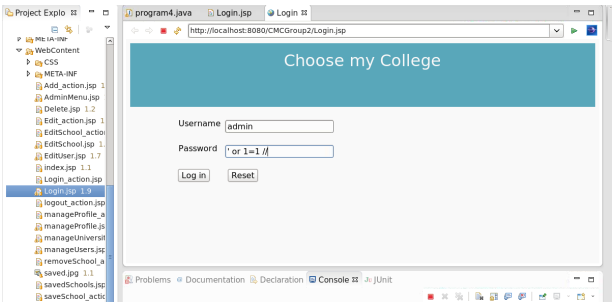

Fig. 6. Choose My College used in CSCI 230, written by Joe Buysee, Sean Deal, and Zach Eich.

My Demonstration for the semester was going to be a injection into the database created for CSCI 230, the Choose My College Website. Upon testing with Strings like ”a or 1=1//” the server had crashed and the ports it needed to access where blocked. After having the server reset I continued to try and manipulate the code so I could access the database, however the server failed again before it could be accessed.

After this I turned to the other website I have had experience with this past semester. For Software Engineering I had been working on a student employment website. Using the same technique, nearly the same

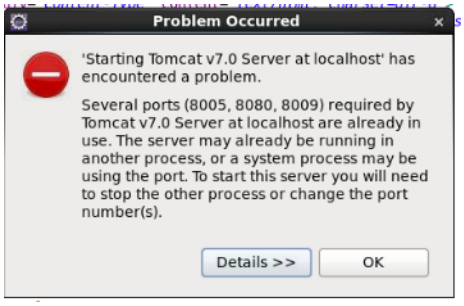

Fig. 7. The port error that kept reoccurring when injecting upon the Choose My College Website. Injections unintentionally actually knocked the site off the server preventing access to the database.

issue occurs. The CSBSJStudent Employment website is hosted by GoDaddy, and therefore much more secure than the one programed for 230. This website when entering the correct password logs in normally. When entering a incorrect non-malicious password it will present a text box implying that the password is incorrect. Then I moved to a malicious password ”x or 1=1–” will actually cause it to go into an infinite loop. It will reset after a few seconds with a completely new error page, claiming a bad connection to the server. From what I understand it effectively queries the database and validates the statement. Since the server continues to process the request I believe it is getting caught up on either of two issues. The first could be that the ”–” serving to comment out the rest of the query is either not commenting out the remaining code. That will confuse the server potentially causing the infinite loop. The other possibility could be that it is commenting out the query however there is more code on the same line that is necessary for the queries execution that is being hidden as well. My only other conclusion is that it does work and wants to log you in however it does not know which user to log you in as. When attempting to get a user from the database the code was to large to fit in the text boxes limited character size. Therefore I could not create a user or pick one that already existed.

VII. Conclusions

While the attacks are becoming more frequent the awareness of defense is not as prevalent for Database defenders. Current methods while not 100% prove effective enough to block the majority of injections on a database, especially the basic injections. While the culture of hacking changes the effects of SQLI can still be felt by Auditors, as it is reintroduced with new techniques. The basis of a Dynamic query still remains the same, and the only way to slow down its popularity is with smarter programming in parametrized statements.

References

[1] Chip Andrews, David Litchfield, and Bill Grindlay. *SQL Server Security*. McGraw Hill, 2003.
[2] Google Articles. New research shows how digital connects shoppers to local stores.
[3] Ericka Chickowski. 10 reasons sql injection still works.
[4] Justin Clarke. *SQL injection attacks and defense*. Syngress Pub., Burlington, MA, 2009.
[5] Zach Eich. *SQL Injection Technical Analysis*.
[6] Jon Foote. Cyber threats to mobile phones.
[7] Steve Friedl. Sql injection attacks by example.
[8] Adam Greenberg. Sql injection attacks still enable breaches, all these years later.
[9] Derrick Harris. 5 predictions on the future of databases (from a guy who knows databases), 2013.
[10] Sean Kerner. Sql injection most dangerous software error.
[11] Arun Kumar. Defacing a website with sql injection explained.
[12] David Litchfield and Books24x7 Inc. The database hacker's handbook defending database servers, 2005.
[13] Stuart McDonald. Sql injection: Modes of attack, defence, and why it matters. Report.
[14] Jay McGregor. The top 5 most brutal cyber attacks of 2014 so far.
[15] N/A. High-tech bridge research: Web application security trends in 2013.
[16] Ellen Nakashima. Researchers identify sophisticated chinese cyberespionage group.
[17] Ron Natan. *Database Security and Auditing*. Elsevier Digital Press, Jordan Hill, 2005.
[18] Frank Ohlhorst. Tips to prevent rising danger from sql injection attacks.
[19] Kevin Poulsen. Guess settles with ftc over cybersecurity snafu.
[20] Online Web Security Project. Owasp top ten 2013.
[21] Robert Trigaux. A history of hacking.
[22] Unknown. Sql injection via xss.
[23] Mark Zuckerburg. Facebook sign up.

Appendix A
Reflections

A. CSCI Classes

Previous Computer science courses have made a large impact on my understanding of the methods of SQL Injection. The class that has had the biggest impact was a class taught by Imad Rahal Computer Organization CSCI 310. I started my Computer Science interest my sophomore year and so my knowledge was limited to what is on the box when you buy a computer. Imads class was one of the most difficult classes I have taken in college but the reward was an understanding of computing and Abstraction in programming that I would not have picked up on my own. This class added C++ to my known languages and helped in visualizing how the computer operates in binary.

Yu Zhangs Algorithms class taught the use of scripting, and shells, which is a common practice especially in the hacking realm. The majority of videos and examples I found were directions from a Terminal to execute code or programs from command line input. While I could not replicate I was able to follow along and understand a lot of what was part of the command line and what was part of the Injection itself.

B. English Classes

Writing is not my strong suite and First Year Seminar along with English 101 gave me a better understanding on how to structure the papers I have submitted. It also helped with class discussions to help build the content of the paper reflecting on ideas and principles I had not come up with. The class room discussions of our papers would always add at least 2 paragraphs to my essays, and the content was well written into the paper.

Then adding to it was 373, where the structure of an abstraction is given new meaning as not just a short introduction, but 5 extremely important statements to draw and hold a readers interest. It holds more content of a paper more than most paragraphs in the paper itself. While this principle will hold for many papers to come it has already shown me the effect a single sentence has on a paper. I also liked the number of presentations we gave in class. While most classes end with a final or a single presentation, we gave multiple helping to build confidence and provide us with a new presenting skill.