MSSQL Identified as Vulnerable to Emoji String Exploitation

MSSQL Identified as Vulnerable to Emoji String Exploitation - DecryptLOL, @decrypt_lol, Decrypt LOL.

- Published: 2024-11-29
- Original: <https://decrypt.lol/posts/2024/11/29/mssql-identified-as-vulnerable-to-emoji-string-exploitation/>
- Preserved from: https://decrypt.lol/posts/2024/11/29/mssql-identified-as-vulnerable-to-emoji-string-exploitation/ (live) on 2026-08-09
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

MSSQL Identified as Vulnerable to Emoji String Exploitation

November 29, 2024 / 3 min read

stories , application security , software vulnerabilities , brute force attack , microsoft sql server , collation bugs

Quick take - Microsoft SQL Server has been found to treat a goblin emoji as equivalent to an empty string, potentially leading to security vulnerabilities in applications that utilize it, particularly in the context of brute-force password attacks.

Fast Facts

- Microsoft SQL Server (MSSQL) treats a goblin emoji as an empty string, creating potential security vulnerabilities for applications using it as a back-end database.
- This inconsistency can lead to significant risks, especially in brute-force password attacks, as it allows exploitation of accounts with blank email addresses.
- The issue stems from discrepancies in Unicode collation logic between MSSQL and application languages, affecting how Unicode characters are handled.
- The vulnerability is not limited to MSSQL; it can also occur in .NET applications on Windows, but does not manifest in .NET Core on Linux.
- Recommendations to mitigate the issue include logging SQL queries, monitoring string comparisons, implementing allow-lists for characters, and conducting thorough security assessments.

Microsoft SQL Server Vulnerability: Goblin Emoji Treated as Empty String

Microsoft SQL Server (MSSQL) has been identified as treating a goblin emoji as equivalent to an empty string. This behavior raises potential security vulnerabilities for applications using MSSQL as their back-end database. The inconsistency in handling strings compared to most application languages can lead to significant security risks, particularly concerning brute-force password attacks.

Analysis of the Vulnerability

Stephen Moir, an Application Security Architect at Pulse Security, brought this issue to light. Moir conducted a detailed analysis of the conditions under which this vulnerability may arise. A test API was developed using .NET Core 8 and Entity Framework to illustrate the problem. This API allows user login via either an email address or username. When a goblin emoji is used as an email address, MSSQL interprets it as an empty string, which can be exploited to conduct brute-force attacks against user accounts with blank email addresses. The vulnerability is particularly concerning as it does not require a valid username, increasing the likelihood of successful attacks.

Discrepancies in Unicode Handling

At the core of the issue is a discrepancy in Unicode collation logic between the application and MSSQL. This discrepancy leads to varied behavior in handling Unicode characters. Testing revealed that not all Unicode characters trigger the same issue. The problem is not confined to MSSQL alone; it can also be encountered in .NET applications running on Windows. Notably, this behavior

does not manifest in .NET Core on Linux, suggesting a possible collation bug specific to the Windows environment. Furthermore, the presence of goblin emojis interspersed with other characters may present additional opportunities for exploitation.

The article underscores the challenges in detecting such vulnerabilities through conventional source code analysis methods. To address the issue, it recommends logging SQL queries and monitoring for string comparisons involving empty strings. Implementing allow-lists for permissible characters and patterns is also advised. The impact of this vulnerability can vary significantly depending on the specific application context, highlighting the need for thorough security assessments. The limitations of “black box” penetration testing in uncovering such vulnerabilities are discussed, advocating for a more comprehensive approach that includes research and reverse engineering techniques. Additionally, the author shares insights into setting up a test API for vulnerability hunting, with the source code for this toy API provided for further reference.

Original Source: Read the Full Article Here