SQL Injection

US-CERT

This paper discusses the Structured Query Language (SQL) injection attack technique and offers mitigation methods.

Overview

Structured Query Language (SQL) injection is an attack technique that attempts to subvert the relationship between a webpage and its supporting database, typically in order to trick the database into executing malicious code. SQL injection usually involves a combination of over-elevated permissions, unsanitized/untyped user input, and/or true software (database) vulnerabilities. Since SQL injection is possible even when no traditional software vulnerabilities exist, mitigation is often much more complicated than simply applying a security patch.

Background

In 2008, there was a significant increase in the number of websites affected by SQL injection attacks. This increase can be attributed in part to the development of automated tools that allowed attackers to test and compromise sites much faster than older manual methods. Eventually, this functionality was included in botnets which used SQL injection attacks to grow the size of the botnet. Another significant contributing factor to this increase is due to the synergy that exists between malware toolkits, such as MPack, and widespread SQL injection attacks. MPack and other malware kits can be configured to exploit a variety of potential vulnerabilities, which increases the likelihood of a successful attack. Conducting SQL injection attacks against a large number of sites increases the number of visitors to the malware kit, which multiplies the return on investment for the attacker.

The following are specific examples of SQL injection events that occurred in 2008:

- April 2008 - attacks against Microsoft Internet Information Services (IIS) that affected more than half a million websites
- December 2008 – Microsoft Internet Explorer 7 (IE7) 0-day exploit that was leveraged via SQL injection attacks

Detection

Detection of SQL injection attacks can be attempted with webserver log auditing combined with network Intrusion Detection Systems (IDS). If you run a production webserver, you must enable logging and periodically review these logs. Numerous software tools exist that allow for the rapid search of webserver logs for specified keywords or regular expressions. Typical searches include commands and characters that should not normally be provided by a user, such as "EXEC", "POST", "UNION", "CAST", or a single quotation mark. While there are many potential searches, each will be useful only in the proper context since valid web forms that require the use of these commands have the potential to generate false positives.

Comprehensive detection of SQL injection attacks is very difficult. Even with logging enabled on all production webservers and Intrusion Detection Systems (IDS) on network chokepoints, the number of methods that attackers can use and combine to evade detection is daunting. For instance, attackers can manipulate the whitespace between commands, encode using decimal, HEX, BASE64, etc., and even inject characters that the webserver / database will ignore in order to evade detection by IDS or log-based analysis.

Prevention is the best approach.

Mitigation/Best Practices

The following mitigation strategies and best practices can be used to minimize the risks associated with this attack vector: As with any system or architecture changes, local administrators are best positioned to know which strategies are appropriate for their specific networks and systems.

Network Level Recommendations

- Deny access to the internet except through proxies for Store and Enterprise servers and workstations.
- Implement firewall rules to block or restrict internet and intranet access for database systems.
- Implement firewall rules to block known malicious IP addresses.
- Harden internal systems against the potential threat posed by a compromised system on the local network. (Do not rely on firewalls to prevent access to insecure systems; secure them.)

System / Application Level Recommendations

- Secure both the operating system and the application.
    - Consider using NIST or other industry standard security checklists to harden both the operating systems and the applications

    - Run only the minimum required applications and services on servers necessary to perform their intended function. In other words, disable all unnecessary applications and services.
    - Follow application vendor security guidelines.
- Update and patch production servers regularly.
    - Include both operating system patches and application patches.
- Disable potentially harmful SQL stored procedure calls.
    - 'xp_cmdshell' on MSSQL has been frequently used by attackers.
- Deny extended URLs.
    - Excessively long URLs can be sent to Microsoft IIS servers, causing the server to fail to log the complete request. Unless specific applications require long URLs, set a limit of 2048 characters. Microsoft IIS will process requests over 4096 bytes long, but will not place the contents of the request in the log files. This has become an effective way to evade detection while performing attacks.
- Sanitize/validate input.
    - Ensure data is properly typed.
    - Ensure data does not contain escaped code.
    - Consider using type-safe stored procedures/prepared statements.
- Ensure error messages are generic and do not expose too much information.
    - Keep error messages short and usable.
    - Do not disclose internal database structure, table names, or account names.
- Use principles of least privilege.
    - Install and run authorized Microsoft SQL Server and IIS services under a non-privileged account.
    - Apply the principle of 'least privilege' on all SQL machine accounts.
    - Remove guest accounts unless operationally necessary.
    - Use an application account for database access.
- Enforce best practice password and account policies.
    - Require the use of a password on Microsoft SQL Server administrator, user, and machine accounts.
    - Change default/built-in account passwords.
    - Change application account passwords regularly.
    - Use strong passwords.
    - Lock out accounts after several unsuccessful logon attempts.
- Document all database accounts, stored procedures, and prepared statements along with their uses.
    - Delete/disable unnecessary accounts (including default accounts).

    - Delete/disable unnecessary stored procedures/prepared statements.
- Perform regular audits and penetration testing.
    - Audit transaction logs for suspicious activity.
    - Audit group and role memberships to ensure enforcement of least access principles.
    - Audit stored procedures on a regular basis and remove unnecessary ones.
    - If you use ASP, consider using Microsoft's code analyzer.
    - Consider using HP's scrawlr utility to help identify problems.
    - Conduct penetration tests against applications, servers, and perimeter security.

Additional Resources

ASP.NET SQLI Prevention
http://msdn.microsoft.com/en-us/library/ms998271.aspx

National Vulnerability Database (NVD) Security Checklists
http://checklists.nist.gov/ncp.cfm?prod_category

Open Web Application Security Project (OWASP) SQLI page:
http://www.owasp.org/index.php/SQL_Injection

Mysql guide to PHP security (ch3 SQL Injection)
http://dev.mysql.com/tech-resources/articles/guide-to-php-security-ch3.pdf

Mysql article on Prepared Statements
http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html