MS-ISAC®
Multi-State Information Sharing & Analysis Center®

APP SEC Advisor

A QUARTERLY APPLICATION SECURITY NEWSLETTER

Vol. 1 / **No. 1** / October 2019
TLP: WHITE

Welcome to the first issue of the AppSec Advisor newsletter!

AppSec Advisor's goal is to communicate to the MS-ISAC community and their peers the best security practices for application design and implementation. We want to achieve confidentiality, integrity, and availability of the data that an application creates, uses, stores, transmits, and disposes.

This newsletter will be a shared responsibility for all the members of the MS-ISAC Application Security workgroup. Please be alert to information that you think may be useful to share with those who will be reading this newsletter; the information you provide may be what a fellow colleague may need. Keep secure!

Contributors

Alder Locke • Senior Development Analyst, Multnomah County, OR
Brett Scott • Applications Security Analyst, Multnomah County, OR
Jacob Bartruff • Senior Development Analyst, Multnomah County, OR
Jessica Cone • Program Specialist, MS-ISAC

Injection Attacks

How to inoculate your web applications from one of the oldest and most dangerous cyberattacks.

What are they? The term "injection" encompasses a range of different attack types. This includes vectors such as SQL injection (the most commonly known type), code injections, LDAP injection, and many more. Depending on the kind of injection different outcomes are possible, such as, unintended data returned, malicious code executed on the users browser or even unwanted commands could be executed on the application's hardware.

Prevalence. Injections are one of the oldest and most dangerous attacks aimed at web applications. This type of threat has continued to be rated the top security risk on OWASP Top 10 Most Critical Web Application Security Risks assessments. This high ranking is due to how dangerous and widespread this type of vulnerability is in applications, especially legacy ones. Another reason for the high ranking is due to how large the attack surface is and how well understood this vulnerability type is. Because of this, many freely available tools exist that allow even the most novice attackers to automatically attack web applications with ease.

Cause. At the core of this type of vulnerability is the injection of untrusted data or code into a trusted environment. For example, if there is a trusted connection between a web application and an underlying database and someone is able to add extra data to a query from the web application then the query can return unintended data or even perform database operations such as deleting data or the database itself. This could be done by modifying the url of the site to include SQL commands or by submitting code into a textbox on the website. This type of attack is a SQL injection.

→ CONTINUED ON NEXT PAGE

Free Tools

Some of these tools may be familiar. Please let the workgroup know of other tools you know that may be useful for future publications.

Wireshark
Popular network protocol analyzer, looks closely at traffic to and from devices (inlc. USB); can save and load captured data files (.pcap). ***AppSec use:*** can help determine which ports and destination IPs an application uses to talk

Nmap
Network discovery and security auditing; can determine what ports are open and services are enabled on remote devices. ***AppSec use:*** can help determine services are enabled by application name and version over which ports

Fiddler
HTTP debugging proxy server app; captures and logs HTTP(S) traffic using man-in-the-middle interception. ***AppSec use:*** can be configured to act as an intermediary for HTTP(S) traffic that an application could use

References

- https://www.troyhunt.com
- https://www.owasp.org/index.php/Main_Page
- https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
- https://www.bsa.org/files/reports/bsa_software_security_framework_web_final.pdf

• • • TERM OF THE MONTH

SAST/DAST–Static (source code) / Dynamic (web app/deployed) Application Security Testing

→ CONTINUED FROM PAGE 1

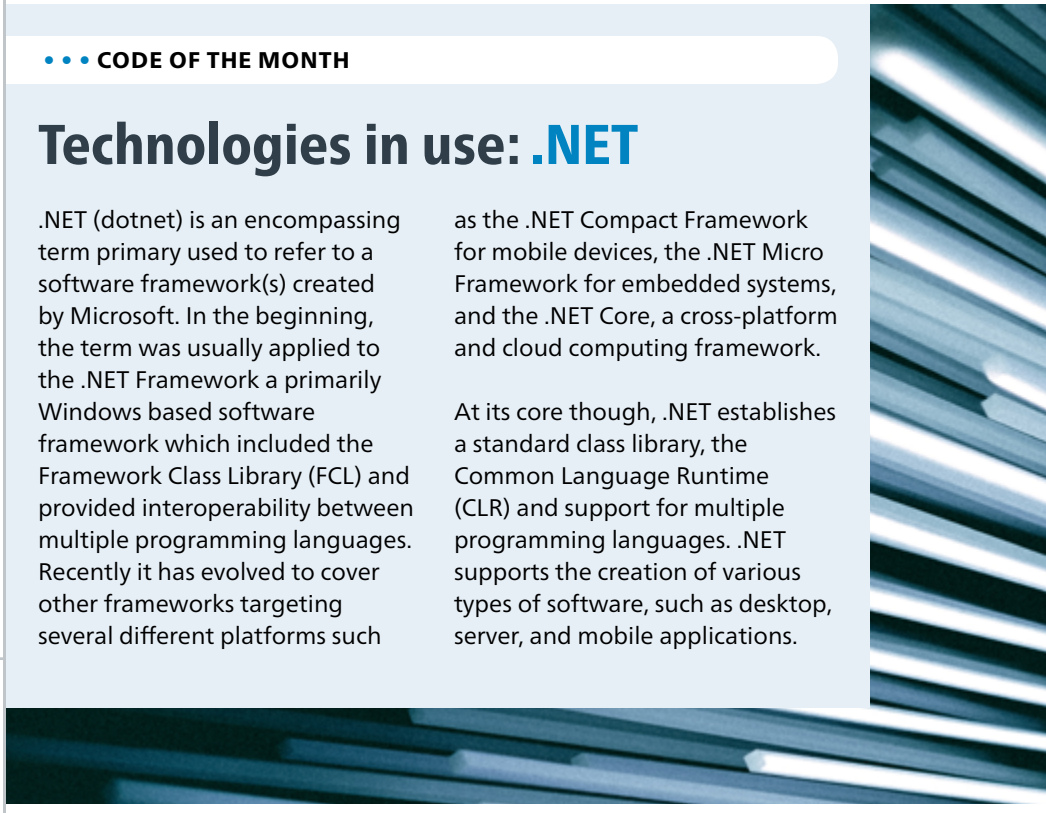
How to mitigate. There are several different ways to mitigate against injection attacks. These include utilizing defenses such as parameterized statements, escaping of user input, input validation, whitelisting, and limiting permissions. Most of these focus around the validation and isolation of how user input is handled by the application, constraining the input into parameters of certain data types or validating that the input is correctly formed. Escaping of user input is another way to limit injection attacks. Limiting the permissions of the credentials used by the application to connect to a database or other system is another good way to mitigate this type of attack. Since the connections between the web application and other systems is often treated as a trusted connection, it is a good idea to isolate what actions the credentials used in these connections can perform on other systems. An example of this would be to remove the ability of a credential to perform actions such as editing a database record or deleting a database when the web application should only perform read actions.

• • • CODE OF THE MONTH

Technologies in use: .NET

.NET (dotnet) is an encompassing term primary used to refer to a software framework(s) created by Microsoft. In the beginning, the term was usually applied to the .NET Framework a primarily Windows based software framework which included the Framework Class Library (FCL) and provided interoperability between multiple programming languages. Recently it has evolved to cover other frameworks targeting several different platforms such as the .NET Compact Framework for mobile devices, the .NET Micro Framework for embedded systems, and the .NET Core, a cross-platform and cloud computing framework.

At its core though, .NET establishes a standard class library, the Common Language Runtime (CLR) and support for multiple programming languages. .NET supports the creation of various types of software, such as desktop, server, and mobile applications.



Resources

- https://en.wikipedia.org/wiki/Code_injection
- https://www.owasp.org/index.php/Injection_Theory
- https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)
- https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- https://dzone.com/articles/what-are-injection-attacks
- https://www.w3schools.com/sql/sql_injection.asp