Code injection

Code injection is the exploitation of a computer bug that is caused by processing invalid data. The injection is used by an attacker to introduce (or "inject") code into a vulnerable computer program and change the course of execution. The result of successful code injection can be disastrous, for example, by allowing computer viruses or computer worms to propagate.

Code injection vulnerabilities occur when an application sends untrusted data to an interpreter. Injection flaws are most often found in SQL, LDAP, XPath, NoSQL queries, OS commands, XML parsers, SMTP headers, program arguments, etc. Injection flaws tend to be easier to discover when examining source code than via testing.[1] Scanners and fuzzers can help find injection flaws.[2]

Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.

Certain types of code injection are errors in interpretation, giving special meaning to user input. Similar interpretation errors exist outside the world of computer science such as the comedy routine Who's on First?. In the routine, there is a failure to distinguish proper names from regular words. Likewise, in some types of code injection, there is a failure to distinguish user input from system commands.

Code injection techniques are popular in system hacking or cracking to gain information, privilege escalation or unauthorized access to a system. Code injection can be used malevolently for many purposes, including:

Arbitrarily modifying values in a database through SQL injection. The impact of this can range from website defacement to serious compromise of sensitive data.

Installing malware or executing malevolent code on a server by injecting server scripting code (such as PHP or ASP).

Privilege escalation to root permissions by exploiting Shell Injection vulnerabilities in a setuid root binary on UNIX, or Local System by exploiting a service on Microsoft Windows.

Attacking web users with HTML/script injection (Cross-site scripting).

In 2008, 5.66% of all vulnerabilities reported that year were classified as Code Injection, the highest year on record. In 2015, this had decreased to 0.77%.[3]

o prevent code injection problems, utilize secure input and output handling, such as:

Using APIs that, if used properly, are secure against all input characters. Parameterized queries (also known as "Compiled queries", "prepared statements", "bound variables") allows for moving user data out of string to be interpreted. Additionally Criteria API[7] and similar APIs move away from the concept of command strings to be created and interpreted.

Enforcing language separation via a static type system.[8]

Input validation, such as whitelisting only known good values, this can be done on client side using JavaScript for example or it can be done on the server side which is more secure.

Input encoding, e.g. escaping dangerous characters. For instance, in PHP, using the htmlspecialchars() function to escape special characters for safe output of text in HTML, and mysqli::real\_escape\_string() to isolate data which will be included in an SQL request, to protect against SQL Injection.

Output encoding, i.e. preventing HTML Injection (XSS) attacks against web site visitors

HttpOnly is a flag for HTTP Cookies that, when set, does not allow client-side script interaction with cookies, thereby preventing certain XSS attacks.[9]

Modular shell disassociation from kernel

With SQL Injection, one can use parameterized queries, stored procedures, whitelist input validation, and more to help mitigate Code Injection problems.[10]

The solutions listed above deal primarily with web-based injection of HTML or script code into a server-side application. Other approaches must be taken, however, when dealing with injection of user code on the user machine, resulting in privilege elevation attacks. Some approaches that are used to detect and isolate managed and unmanaged code injections are:

Runtime image hash validation – capture a hash of a part or complete image of the executable loaded into memory, and compare it with stored and expected hash.

NX bit – all user data is stored in a special memory sections that are marked as non-executable. The processor is made aware that no code exists in that part of memory, and refuses to execute anything found in there.

Canaries – randomly place values in a stack. At runtime, a canary is checked when a function returns. If a canary has been modified, the program stops execution and exits. This occurs on a Stack Overflow Attack.

[In C]Code Pointer Masking (CPM) – after loading a (potentially changed) code pointer into a register, apply a bitmask to the pointer. This effectively restricts the addresses to which the pointer can refer.