Software Assurance Tips

A product of the Software Assurance Tips Team[3]

Jon Hood

Monday 29th November, 2021

1 Malicious Injection of Source Code

Updated Wednesday 11th February, 2026

Suppose that you are a spy, tasked with embedding a malicious backdoor into enemy source code. After being hired onto the team, you find that every line of code is subject to manual code reviews. One of the best ways to hide a Trojan is in plain sight: directly in the source code that's being reviewed.

1.1 Unicode Injections

One of my favorite types of attacks is the homoglyph attack.[2] Developers can embed a function so deeply into code with the apparent same name as a benign-looking version of the function. Consider the homoglyphs used in Boucher and Anderson's example in Figure 1.[1] By using a Cyrillic `H` and hiding the malicious `sayHello()` deep into the code, the developers could be tricked into thinking that a different function is being called.

```
#include <iostream>
void sayHello() {
        std::cout << "Hello, World!\n";
}
void sayHello() {
        std::cout << "Goodbye, World!\n";
}
int main() {
        sayHello();
        return 0;
}
```

Listing 1: Homoglyphic Function

But my favorite type of injection involves the use of bidirectional (BIDI) unicode symbols. Using these symbols, the order of *display* can be changed from the order of *compiler evaluation*. Consider the function in Figure 2. While functions or a return value may appear to be commented out on a web browser or development IDE, they are actually part of the code and become a sneaky way to inject logic that appears commented out to a reviewer.

```
#include <iostream>

bool isAdmin()
{
        /* If we are an admin,  */ return true ;
        std::cerr << "You are not an admin." << std::endl;
        return false;
}

int main()
{
        if (isAdmin())
        {
                std::cout << "You are an admin." << std::endl;
        }
        else
        {
                std::cout << "You are NOT an admin." << std::endl;
        }
        return 0;
}
```

Listing 2: Comment Reordering

References

[1] Nicholas Boucher and Ross Anderson. “Trojan Source: Invisible Vulnerabilities”. In: Preprint (2021). arXiv: 2111.00169 [cs.CR]. URL: https://arxiv.org/abs/2111.00169.

[2] Jon Hood. “Homoglyphs and Homographic Attacks”. In: SwATips.com (2021). URL: https://www.swatips.com/articles/20210510.html.

[3] Jon Hood, ed. SwATips. https://www.SwATips.com/.