CVE-2019-13225

Referaat aines “Andmeturve”

Koostaja: Siim Karel Koger
Juhendaja: Meelis Roos

Tartu 2020
Käesolevaga luban Tartu Ülikoolil seda referaati avalikult eksponeerida kuni aastani 2025.
Siim Karel Koger 29.04.2020

Introduction 3
Vulnerability description 4
Vulnerability fix 6
References 8
Introduction

Oniguruma is a modern and flexible regular expressions library. It encompasses features from different regular expression implementations that traditionally exist in different languages. Oniguruma is a BSD licensed library that supports a variety of character encodings. The Ruby programming language, in version 1.9, as well as PHP's multi-byte string module (since PHP5), use Oniguruma as their regular expression engine.

A NULL Pointer Dereference in Oniguruma 6.9.2 allows attackers to potentially cause a denial of service by providing a crafted regular expression. The only thing a user can do according to security websites is to upgrade to a newer version 6.9.3. It may be possible to downgrade to 5.X.X as per author’s words “5.X.X does not have if-then-else pattern (?(cond)then|else) feature. This bug fix is about implementation of the if-then-else pattern, so it has nothing to do with 5.X.X.”, however in that case other, already fixed bugs, may resurface for the user.
Vulnerability description

NULL pointer dereference is a case where a pointer is dereferenced but it doesn’t point to any valid object. This can cause a program to crash and cause a denial of service. There are no complete fixes for NULL pointer dereferences aside from conscientious programming, but there are steps you can take so that it doesn’t occur. Although Oniguruma is written in C++, the simple steps mentioned here can be used to avoid this or similar problems in any mainstream language.

For example, in this following case, the function takes an IP address from a user, verifies that it’s valid, looks up the hostname, and copies it into a buffer. A problem arises when an attacker provides an address that looks valid but does not actually resolve to a hostname - in that case, `gethostbyaddr(addr, sizeof(struct in_addr), AF_INET)` will return NULL and code doesn’t check variable `hp`, so a NULL pointer dereference would occur in the call to `strcpy(hostname, hp -> h_name)`.

```c
void host_lookup(char *user_supplied_addr){
    struct hostent *hp;
    in_addr_t *addr;
    char hostname[64];
    in_addr_t inet_addr(const char *cp);

    /*routine that ensures user_supplied_addr is in the right format for conversion */
    validate_addr_form(user_supplied_addr);
    addr = inet_addr(user_supplied_addr);
    hp = gethostbyaddr( addr, sizeof(struct in_addr), AF_INET);
    strcpy(hostname, hp->h_name);
}
```

A simple and effective way to avoid an error in such cases is to just check if the pointer is NULL or not before using the value.
if (pointer1 != NULL) {
/* make use of pointer1 */
}

One must remember though - if a multithreaded or other asynchronous environment is worked with, you should ensure that lock is used and the shared resource (pointer) is being accessed synchronously, or else there might be a case where you check that the pointer is not NULL on one thread, change it with another, and then use the value on the first thread.
Vulnerability fix

The issue resided on the way 'If/Else' statements were handled by Oniguruma (file src/regcomp.c, method static int compile_bag_node(BagNode* node, regex_t* reg, ScanEnv* env)). The method should return a reference to some desired variable (not important in this context). The problem arose in the last part of the method inside “case BAG_IF_ELSE:”. Before the fix, the last if-statement is entered when “Else” is not null which is correct, however, in that statement, there is an “add_op()” method call to set the value of “r” (which is returned in the end). If “Else” is NULL and “r” is not set before this if-statement then the method returns a NULL which then causes the error wherever this method return value is dereferenced (also not important where - anywhere).

In an updated version (commit c509265c5f6ae7264f7b8a8aae1cfa5fc59d108c) two add_op() calls were pulled out of the if statement to return those values if they are not 0.
if (IS_NOT_NULL(Else)) {
    else_len = compile_length_tree(Else, reg);
    if (else_len < 0) return else_len;
}
else
    else_len = 0;

r = add_op(reg, OP_JUMP);
if (r != 0) return r;
COP(reg)->jump.addr = SIZE_OP_ATOMIC_END + else_len + SIZE_INC_OP;

r = add_op(reg, OP_ATOMIC_END);
if (r != 0) return r;

if (IS_NOT_NULL(Else)) {
    r = compile_tree(Else, reg, env);
}
}
References

5. https://security.gentoo.org/glsa/201911-03
7. https://github.com/kkos/oniguruma
8. https://www.w3schools.com/cpp/cpp_pointers_dereference.asp