ENV02-C. Beware of multiple environment variables with the same effective name

The `getenv()` function searches an environment list for a string that matches a specified name and returns a pointer to a string associated with the matched list member.

Subclause 7.22.4.6 of the C Standard [ISO/IEC 9899:2011] states:

> The set of environment names and the method for altering the environment list are implementation-defined.

Depending on the implementation, multiple environment variables with the same name may be allowed and can cause unexpected results if a program cannot consistently choose the same value. The GNU glibc library addresses this issue in `getenv()` and `setenv()` by always using the first variable it encounters and ignoring the rest. However, it is unwise to rely on this behavior.

One common difference between implementations is whether or not environment variables are case sensitive. Although UNIX-like implementations are generally case sensitive, environment variables are "not case sensitive in Windows 98/Me and Windows NT/2000/XP" [MSDN].

Duplicate Environment Variable Detection (POSIX)

The following code defines a function that uses the POSIX `environ` array to manually search for duplicate key entries. Any duplicate environment variables are considered an attack, so the program immediately terminates if a duplicate is detected.
extern char **environ;

int main(void) {
    if (multiple_vars_with_same_name()) {
        printf("Someone may be tampering.\n");
        return 1;
    }

    /* ... */
    return 0;
}

int multiple_vars_with_same_name(void) {
    size_t i;
    size_t j;
    size_t k;
    size_t l;
    size_t len_i;
    size_t len_j;

    for(size_t i = 0; environ[i] != NULL; i++) {
        for(size_t j = i; environ[j] != NULL; j++) {
            if (i != j) {
                k = 0;
                l = 0;

                len_i = strlen(environ[i]);
                len_j = strlen(environ[j]);

                while (k < len_i && l < len_j) {
                    if (environ[i][k] != environ[j][l])
                        break;
                    if (environ[i][k] == '=')
                        return 1;
                    k++;
                    l++;
                }
            }
        }
    }
    return 0;
}

Noncompliant Code Example

This noncompliant code example behaves differently when compiled and run on Linux and Microsoft Windows platforms:

```c
if (putenv("TEST_ENV=foo") != 0) {
    /* Handle error */
}
if (putenv("Test_ENV=bar") != 0) {
    /* Handle error */
}
const char *temp = getenv("TEST_ENV");
if (temp == NULL) {
    /* Handle error */
}
printf("%s\n", temp);
```

On an IA-32 Linux machine with GCC 3.4.4, this code prints
whereas, on an IA-32 Windows XP machine with Microsoft Visual C++ 2008 Express, it prints

Compliant Solution

Portable code should use environment variables that differ by more than capitalization:

```c
if (putenv("TEST_ENV=foo") != 0) {
    /* Handle error */
}
if (putenv("OTHER_ENV=bar") != 0) {
    /* Handle error */
}
const char *temp = getenv("TEST_ENV");
if (temp == NULL) {
    /* Handle error */
}
printf("%s\n", temp);
```

Risk Assessment

An attacker can create multiple environment variables with the same name (for example, by using the POSIX `execve()` function). If the program checks one copy but uses another, security checks may be circumvented.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENV02-C</td>
<td>Low</td>
<td>Unlikely</td>
<td>Medium</td>
<td>P2</td>
<td>L3</td>
</tr>
</tbody>
</table>

Automated Detection

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Checker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compass/ROSE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parasoft C/C++test</td>
<td>10.4.2</td>
<td>CERT_C-ENV02-a</td>
<td>Usage of system properties (environment variables) should be restricted</td>
</tr>
</tbody>
</table>

Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

<table>
<thead>
<tr>
<th>SEI CERT C++ Coding Standard</th>
<th>VOID ENV00-CPP. Beware of multiple environment variables with the same effective name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO/IEC TR 24772:2013</td>
<td>Executing or Loading Untrusted Code [XYS]</td>
</tr>
<tr>
<td>MITRE CWE</td>
<td>CWE-462, Duplicate key in associative list (Alist)</td>
</tr>
<tr>
<td></td>
<td>CWE-807, Reliance on untrusted inputs in a security decision</td>
</tr>
</tbody>
</table>

Bibliography

| MSDN                        | getenv()                                                                                 |