Use of inherently dangerous function

Name: Use of inherently dangerous function

Description: Using a library function that does not check buffer bounds requires the surrounding program to be very carefully written to avoid buffer overflows.

ID: cpp/potential-buffer-overflow

Kind: problem

Severity: warning

This rule highlights potentially overflowing calls to the functions `sprintf`, `vsprintf`, and `gets` with a warning. These functions allow unbounded writes to buffers, which may cause an overflow when used on untrusted data or without adequate checks on the size of the data. Function calls of this type constitute a security risk through buffer overflows. The `gets` function, in particular, is one of the vulnerabilities exploited by the Internet Worm of 1988, one of the first computer worms to spread through the Internet.

Recommendation

Always control the length of buffer copy and buffer write operations. Use the safer variants `snprintf`, `vsnprintf`, and `fgets`, which include an extra buffer length argument.

Example

```
 1  void f(char* s, float f) {
 2      char buf[30];
 3
 4      //wrong: gets has no limit to the length of data it puts in the buffer
 5      gets(buf);
 6
 7      //wrong: sprintf does not limit the length of the string put into buf
 8      sprintf(buf, "This is a string: %s", s);
 9
10      //wrong: %f can expand to a very long string in extreme cases, easily overrunning this
buffer
11      sprintf(buf, "This is a float: %f", f);
12  }
```

To improve the security of this example code, three changes should be made:

1. Introduce a preprocessor define for the size of the buffer.
2. Replace the call to `gets` with `fgets`, specifying the define as the maximum length to copy. This will prevent the buffer overflow.
3. Replace both calls to `sprintf` with `snprintf`, specifying the define as the maximum length to copy. This will prevent the buffer overflow.
4. Consider using the %g format specifier instead of %f.

References

- Common Weakness Enumeration: CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow').
- CERT C Coding Standard: STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator.
- M. Howard, D. Leblanc, J. Viega, *19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*, McGraw-Hill Osborne, 2005.
- Wikipedia: Morris worm.

- E. Spafford. *The Internet Worm Program: An Analysis*. Purdue Technical Report CSD-TR-823, (online) , 1988.
- Common Weakness Enumeration: CWE-242.