6.64 Reliance on external format string [SHL]

6.64.1 Description of application vulnerability

Many languages use format string to control how output is generated or input acquired. If the format string can be influenced by an attacker, there is an opportunity for them to gain access to what should be private data and to execute arbitrary code.

Deleted: The software uses externally controlled format strings in input/output functions, which can lead to buffer overflows or data representation problems.

Comment [CP1]: I have to say that I'm not entirely convinced by this claim. The CWE doesn't provide an example and what you can write back in C/C++/Perl is very limited. There is a rather contrived example of how there could be a security issue (modifying a 'checked' flag), but not of code execution (but see my second comment)

Deleted:

6.64.2 Cross reference

CWE:

134. Uncontrolled Format String

6.64.3 Mechanism of failure

There are three common scenarios where a format string can be a source of vulnerability. In all cases a key factor is that the format string is acquired from some source outside the program's control, and which may have been supplied or modified by an attacker:

Deleted: The programmer rarely intends for a format string to be user-controlled at all. This weakness frequently occurs in code that constructs log messages, where a constant format string is omitted. ... [1]

- the supplied format string may specify extremely large field widths for output, potentially leading to resource exhaustion and ultimately program failure
- where the number of parameters an output statement expects is controlled by the contents of the format string, if the number of items to be output according to the format string is greater than the number of parameters supplied by the program, then the statement will start outputting arbitrary data from the stack, which may be sensitive
- most control sequences in format strings cause a supplied parameter to be read and generate appropriate output. However, some languages allow for control sequences that write a value (typically the number of characters output so far) to the supplied pointer parameter. Using such a control sequences in a tainted format string may lead to unexpected modification of the program

6.64.4 Applicable language characteristics

This vulnerability is intended to be applicable to languages with the following characteristics:

- Languages that support format strings for input/output functions.

6.64.5 Avoiding the vulnerability or mitigating its effects

Software developers can avoid the vulnerability or mitigate its ill effects in the following ways:

Comment [CP2]: If the print statement is an sprintf, I can clearly write arbitrary code into the string buffer. If I can I then force it to be executed, then there is a way of executing arbitrary code

- Where possible, ensure that all format strings are supplied to their functions as static strings which cannot be modified by an attacker. During development (i.e. not at run time), it should be checked that:
  - the number of arguments supplied to the function matches those required by the format string.
  - all specifiers used match the associated parameter.

Deleted: functions are passed as

Deleted: controlled by the user

Deleted: and that

Deleted: proper number of arguments is always sent to that function

Deleted: Ensure

-
- If a format string has to be supplied from outside the program, i.e. not as a static string, the program should check at run-time:
    - the number of parameters required by the format string matches the number supplied in the call
    - the field widths required by the format string are not excessive
    - no use is made of control structures that modify the program's data

Deleted: Avoid format strings that will write to a memory location that is pointed to by its argument.

6.64.6 Implications for language design and evolution

In future language design and evolution activities, the following items should be considered:

- Ensure all format strings are verified to be correct in regard to the associated arguments or parameters, either at compile time (for static strings) or at run-time, if the format string is acquired from an external source

Deleted: .

Page 1: [1] Deleted	Clive Pygott	7/20/17 6:24:00 PM

The programmer rarely intends for a format string to be user-controlled at all. This weakness frequently occurs in code that constructs log messages, where a constant format string is omitted.

In cases such as localization and internationalization, the language-specific message repositories could be an avenue for exploitation, but the format string issue would be resultant, since attacker control of those repositories would also allow modification of message length, format, and content.