# Chapter 56

# perlsec

Perl security

## 56.1 DESCRIPTION

Perl is designed to make it easy to program securely even when running with extra privileges, like setuid or setgid programs. Unlike most command line shells, which are based on multiple substitution passes on each line of the script, Perl uses a more conventional evaluation scheme with fewer hidden snags. Additionally, because the language has more builtin functionality, it can rely less upon external (and possibly untrustworthy) programs to accomplish its purposes.

Perl automatically enables a set of special security checks, called *taint mode*, when it detects its program running with differing real and effective user or group IDs. The setuid bit in Unix permissions is mode 04000, the setgid bit mode 02000; either or both may be set. You can also enable taint mode explicitly by using the **-T** command line flag. This flag is *strongly* suggested for server programs and any program run on behalf of someone else, such as a CGI script. Once taint mode is on, it's on for the remainder of your script.

While in this mode, Perl takes special precautions called *taint checks* to prevent both obvious and subtle traps. Some of these checks are reasonably simple, such as verifying that path directories aren't writable by others; careful programmers have always used checks like these. Other checks, however, are best supported by the language itself, and it is these checks especially that contribute to making a set-id Perl program more secure than the corresponding C program.

You may not use data derived from outside your program to affect something else outside your program–at least, not by accident. All command line arguments, environment variables, locale information (see *perllocale*), results of certain system calls (readdir(), readlink(), the variable of shmread(), the messages returned by msgrcv(), the password, gcos and shell fields returned by the getpwxxx() calls), and all file input are marked as "tainted". Tainted data may not be used directly or indirectly in any command that invokes a sub-shell, nor in any command that modifies files, directories, or processes, **with the following exceptions**:

- Arguments to `print` and `syswrite` are **not** checked for taintedness.
- Symbolic methods

  ```
  $obj->$method(@args);
  ```

  and symbolic sub references

  ```
  &{$foo}(@args);
  $foo->(@args);
  ```

  are not checked for taintedness. This requires extra carefulness unless you want external data to affect your control flow. Unless you carefully limit what these symbolic values are, people are able to call functions **outside** your Perl code, such as POSIX::system, in which case they are able to run arbitrary external code.