Untrusted input for a condition

Name: Untrusted input for a condition

Description: Using untrusted inputs in a statement that makes a security decision makes code vulnerable to attack.

ID: cpp/tainted-permissions-check

Kind: problem

Severity: warning

Precision: medium

This rule finds code where untrusted inputs are used in an `if` statement, and the body of that statement makes a security decision. This is an example of CWE-807 and makes the program vulnerable to attack. An attacker might be able to gain unauthorized access to the system by manipulating external inputs to the system.

Recommendation

In most cases, you need to add or strengthen the checks made on the user-supplied data to ensure its integrity. The user-supplied data can then be used as a trusted input to the security decision. For example, instead of checking an HTTP cookie against a predictable fixed string, check a cookie against a randomly generated session key.

This rule may highlight a few conditions where user-supplied data has been checked and can be trusted. It is not always possible to determine if the checks applied to data are enough to ensure security.

Example

The following example is included in CWE 807.

```
1  struct hostent *hp;struct in_addr myaddr;
2  char* tHost = "trustme.example.com";
3  myaddr.s_addr=inet_addr(ip_addr_string);
4
5  hp = gethostbyaddr((char *) &myaddr, sizeof(struct in_addr), AF_INET);
6  if (hp && !strncmp(hp->h_name, tHost, sizeof(tHost))) {
7    trusted = true;
8  } else {
9    trusted = false;
10 }
```

In this example, the result of a reverse DNS query is compared against a fixed string. An attacker can return an incorrect reverse DNS entry for the requesting IP and thus gain the same access as a legitimate user from `trustme.example.com`.

To fix the problem in this example, you need to add an additional mechanism to test the user-supplied data. For example, numeric IP addresses could be used.

References

- Common Weakness Enumeration: CWE-807.