Conditionally uninitialized variable

Name: Conditionally uninitialized variable

Description: An initialization function is used to initialize a local variable, but the returned status code is not checked. The variable may be left in an uninitialized state, and reading the variable may result in undefined behavior.

ID: cpp/conditionally-uninitialized-variable

Kind: problem

Severity: warning

A common pattern is to initialize a local variable by calling another function (an "initialization" function) with the address of the local variable as a pointer argument. That function is then responsible for writing to the memory location referenced by the pointer.

In some cases, the called function may not always write to the memory pointed to by the pointer argument. In such cases, the function will typically return a "status" code, informing the caller as to whether the initialization succeeded or not. If the caller does not check the status code before reading the local variable, it may read unitialized memory, which can result in unexpected behavior.

Recommendation

When using a initialization function that does not guarantee to initialize the memory pointed to by the passed pointer, and returns a status code to indicate whether such initialization occurred, the status code should be checked before reading from the local variable.

Example

In this hypothetical example we have code for managing a series of devices. The code includes a `DeviceConfig` struct that can represent properties about each device. The `initDeviceConfig` function can be called to initialize one of these structures, by providing a "device number", which can be used to look up the appropriate properties in some data store. If an invalid device number is provided, the function returns a status code of `-1`, and does not initialize the provided pointer.

In the first code sample below, the `notify` function calls the `initDeviceConfig` function with a pointer to the local variable `config`, which is then subsequently accessed to fetch properties of the device. However, the code does not check the return value from the function call to `initDeviceConfig`. If the device number passed to the `notify` function was invalid, the `initDeviceConfig` function will leave the `config` variable uninitialized, which will result in the `notify` function accessing uninitialized memory.

```
1  struct DeviceConfig {
2    bool isEnabled;
3    int channel;
4  };
5
6  int initDeviceConfig(DeviceConfig *ref, int deviceNumber) {
7    if (deviceNumber >= getMaxDevices()) {
8      // No device with that number, return -1 to indicate failure
9      return -1;
10   }
11   // Device with that number, fetch parameters and initialize struct
12   ref->isEnabled = fetchIsDeviceEnabled(deviceNumber);
13   ref->channel = fetchDeviceChannel(deviceNumber);
14   // Return 0 to indicate success
15   return 0;
16 }
```

```

int notify(int deviceNumber) {
  DeviceConfig config;
  initDeviceConfig(&config, deviceNumber);
  // BAD: Using config without checking the status code that is returned
  if (config.isEnabled) {
    notifyChannel(config.channel);
  }
}
```

To fix this, the code needs to check that the return value of the call to `initDeviceConfig` is zero. If that is true, then the calling code can safely assume that the local variable has been initialized.

```
struct DeviceConfig {
  bool isEnabled;
  int channel;
};

int initDeviceConfig(DeviceConfig *ref, int deviceNumber) {
  if (deviceNumber >= getMaxDevices()) {
    // No device with that number, return -1 to indicate failure
    return -1;
  }
  // Device with that number, fetch parameters and initialize struct
  ref->isEnabled = fetchIsDeviceEnabled(deviceNumber);
  ref->channel = fetchDeviceChannel(deviceNumber);
  // Return 0 to indicate success
  return 0;
}

void notify(int deviceNumber) {
  DeviceConfig config;
  int statusCode = initDeviceConfig(&config, deviceNumber);
  if (statusCode == 0) {
    // GOOD: Status code returned by initialization function is checked, so this is safe
    if (config.isEnabled) {
      notifyChannel(config.channel);
    }
  }
}
```

References

- Wikipedia: Uninitialized variable.
- Common Weakness Enumeration: CWE-457.