

Forum:[Hack Forums](#)

Thread: [AV Detection & Runtime Security - Detecting Unknown DLLs in C++](#)

Hackers - March 3, 2024, 11:25 am

Hi all, this is a code snippet I will share that works for detecting any DLL files that are loaded into your script. This will automatically exit the program if an unknown DLL is found, but you can modify it to do whatever you need in that case. This thread is titled about AV bypasses to get your attention, but it's truly not about trying to hide from an AV. It's about making sure NO unknown DLLs can be injected into your program - so it's not about trying to do anything bad, it's about keeping your software safe for your intellectual property, user data, and more.

<https://github.com/LNodesL/Detect-Uknown-DLL-in-Runtime-CPP-Single-File-Header-Include>

```
Code #ifndef UNKNOWNDLL_H
#define UNKNOWNDLL_H
```

```
#include <Windows.h>
#include <Psapi.h>
#include <iostream>
#include <vector>
#include <string>
```

```
// Function to get the file name of the current executable
std::string GetExecutableFileName() {
    char szFileName[MAX_PATH];
    GetModuleFileNameA(NULL, szFileName, MAX_PATH);
    return szFileName;
}
```

```
// Function to enumerate loaded modules and detect unknown DLLs
void DetectUnknownModules() {
    DWORD processId = GetCurrentProcessId();
    HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATION |
PROCESS_VM_READ, FALSE, processId);
    if (hProcess == NULL) {
        std::cerr << "Failed to open process." << std::endl;
        return;
    }
```

```
HMODULE hModules[1024];
DWORD cbNeeded;
if (EnumProcessModules(hProcess, hModules, sizeof(hModules), &cbNeeded)) {
    int moduleCount = cbNeeded / sizeof(HMODULE);
    std::vector<std::string> knownModules = {
        "ntdll.dll",
        "KERNEL32.DLL",
        "KERNELBASE.dll",
        "apphelp.dll",
```

```

    "ucrtbase.dll"
};

// Add the current executable to the list of known modules
knownModules.push_back(GetExecutableFileName());

for (int i = 0; i < moduleCount; i++) {
    char szModName[MAX_PATH];
    if (GetModuleFileNameExA(hProcess, hModules[i], szModName, sizeof(szModName))) {
        std::string moduleName = szModName;
        bool isKnownModule = false;
        for (const auto& knownModule : knownModules) {
            if (moduleName.find(knownModule) != std::string::npos) {
                isKnownModule = true;
                break;
            }
        }
        if (!isKnownModule) {
            std::cout << "Unknown DLL loaded: " << moduleName << std::endl;
            exit(0);
            // Handle detection of unknown module
        }
    }
}
} else {
    std::cerr << "Failed to enumerate process modules." << std::endl;
}

CloseHandle(hProcess);
}

```

#endif // UNKNOWNDLL_H

We have a list of common DLLs, but you should only add DLLs that you trust & are using in your real program for that list.

The next part adds the current executable to the list, but keep in mind that certain VMs won't have this function available so this could error if there is no way to get the current executable name. It also would fail if they split the current executable into different processes, so keep that in mind. Obviously if the goal is to exit the program if it's being injected (into by another program) then a hard crash on a bad executable name lookup might not be a concern to you.

Feel free to leave a reply if you have any thoughts, questions, etc.

webdev77 - March 3, 2024, 11:30 am

Checking the name of a DLL is not a secure method. You need to check the signatures and file hashes to see if they match known versions. I could simply name a malicious DLL to ntdll.dll or similar and it would slip right through.

Hackers - March 3, 2024, 12:10 pm

(Mar 3, 2024 12:30 PM)webdev77 Wrote: [CITE](#) Checking the name of a DLL is not a secure method. You need to check the signatures and file hashes to see if they match known versions. I could simply name a malicious DLL to ntdll.dll or similar and it would slip right through.

That is very true, and a good point. I do plan on expanding on this code in the future to be more of a C++ class with a few different functions, so will keep this in mind because we can do named

searches as well as finding different signatures for kernel calls, api uses, etc. I am doing some PE deconstruction and haven't yet fully disassembled a dll besides loading and dynamically finding functions for ntdll, but I suppose possibly one way to handle this would be to compare known type definitions, dwords, and other variable or struct types to the disassembled programs we're detecting. This would then allow us to detect programs that are using wrapper functions or reflections of other tools & APIs - perhaps for example detecting something that is setup to work with specific syscalls or minifilters. This could be a way to dynamically detect different behaviors and signatures of a dll or program running alongside our software to prevent any malicious use against our program, analysis, etc.

Stanley - *March 16, 2026, 8:21 am*

I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley

yigliman - *March 30, 2026, 9:16 am*

I'd like to know how to obfuscate my .exe file or how to make it undetectable by antivirus software.

Vladimir_Belyakov - *July 18, 2026, 3:07 am*

It is important to know which process your EXE file runs and which antivirus the system is using.