t Registry Callbacks - Registry Writes Withou

deceptiq.com/blog/ntuser-man-registry-persistence

Rad Kawar December 27, 2025

Registry Writes Without Registry Callbacks

Registry persistence remains one of the most reliable ways to survive a reboot. Run keys, shell extensions, COM hijacks: the patterns are well documented and the detection coverage reflects this.

This post explores a technique for establishing registry persistence and registry writes against HKCU at medium integrity without triggering registry callbacks.

EDR solutions monitor registry modifications through `CmRegisterCallbackEx`. This kernel callback mechanism allows drivers to intercept registry operations before they complete. When a process calls `RegSetValue` or `RegCreateKey`, registered callbacks receive notification with full context: the key path, the data being written, and the process responsible.

Kernel Patch Protection (PatchGuard) prevents vendors from hooking the kernel directly. `CmRegisterCallbackEx` provides a supported alternative, and most endpoint security products rely on it for registry visibility.

Elastic's Uncommon Registry Persistence Change rule is representative. It monitors registry paths including:

```
1HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

2HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit

3HKLM\SYSTEM\ControlSet*\Control\Session Manager\BootExecute

4HKEY_USERS\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\Load

5HKEY_USERS\*\Environment\UserInitMprLogonScript
```

These paths represent classic persistence locations. The detection logic filters out known legitimate modifications and alerts on the rest.

The approach works because registry writes go through the registry APIs, which trigger the callbacks, which generate telemetry. Every step in this chain is well instrumented.

The chain assumes registry writes use the registry APIs.

Windows user profiles store registry hives in the user's profile directory. `NTUSER.DAT` contains the user's `HKEY_CURRENT_USER` hive and gets loaded at logon. `NTUSER.MAN` takes priority when present.

The `.MAN` extension denotes a *mandatory profile*. Originally designed for kiosk and shared workstation scenarios, Windows checks for this file first and loads it instead of the standard `.DAT` hive. Microsoft's documentation describes the intended use case:

> A mandatory user profile is a roaming user profile that has been preconfigured by an administrator to specify settings for users.

The Active Directory Migration Tool (ADMT) provides an example of this priority handling. When translating user profiles during domain migrations, ADMT must locate the correct hive file. The source code shows the priority logic:

```
1// look for NTUser.MAN

2_snwprintf(profileName, profileNameBufferSize, L"%s\\NTUser.MAN", targetPath);

3profileName[profileNameBufferSize - 1] = L'\0';

4hFind = FindFirstFile(profileName, &fDat);

5if (hFind != INVALID_HANDLE_VALUE)

6{

7    err.MsgWrite(0, DCT_MSG_TRANSLATING_NTUSER_MAN_S, targetAcctName);

8    rc = TranslateUserProfile(...);
```

```
9    FindClose(hFind);
10}
11else
12{
13    // check for NTUser.DAT
14    _snwprintf(profileName, profileNameBufferSize, L"%s\\NTUser.DAT", targetPath);
15    // ...
16}
```

When `NTUSER.MAN` exists, Windows does not merge it with `NTUSER.DAT`. The `.MAN` file becomes the entire HKCU hive for that session.

Placing a crafted `NTUSER.MAN` in `C:\Users\<target>\` loads persistence keys into HKCU on next logon. The hive is loaded directly from disk without invoking registry APIs.

`CmRegisterCallbackEx` monitors registry operations. Hive loads are not registry operations. The callbacks are not invoked.

Filesystem events will trigger. Writing the file to the profile directory is visible to any EDR monitoring file operations. Registry-focused detections remain blind.

Medium integrity is sufficient. The target is the user's own profile directory, which they have write access to. Local admin is not required.

The profile subsystem loads `NTUSER.MAN` *instead of* `NTUSER.DAT`, not alongside it. The target user's existing hive is required as a starting point, with persistence keys added to it. Enterprise environments configure application settings, shell preferences, environment variables, and policy state through the registry. Replacing the hive wholesale results in a visibly broken session.

The registry can be exported in two formats: binary (the raw hive) or `.reg` (text format). Binary export requires elevation. Text export does not.

The general recommended workflow, while you can drop an arbitrary hive on disk, is as follows:

1. Export the target user's HKCU hive as `.reg` format (BOFs exist for this)
2. Add persistence entries to the `.reg` file
3. Convert to binary hive format using tooling such as HiveSwarming

4. Write the modified hive as `NTUSER.MAN` in the target's `%USERPROFILE%` directory
5. Wait for logon

On next login, Windows loads the hive from `NTUSER.MAN` with your additional registry keys now present in HKCU - without registry API calls.

User hives remain locked while the session is active; Windows holds a handle on the loaded hive, preventing modification or replacement.

`NTUSER.MAN` cannot be swapped while the target user is logged in. Activation requires logoff and logon, or a system reboot. This makes it a persistence mechanism rather than immediate execution.

Operating as a different user or SYSTEM allows staging the file in the target's profile directory, but the hive will not load until their next session starts. Removal works the same way: log in as a different user so the hive is not locked.

Mandatory profiles can be assigned via Active Directory. The profile path attribute on a user object points to a network share:

```
1\\server\share\profile.v6
```

This opens two possible avenues:

1. Modify the profile path. With write access to the target user's AD object, point their profile path to a share you control containing a crafted `NTUSER.MAN`.

On next logon, Windows fetches the hive from your share. This requires Write Property on the `profilePath` attribute - typically granted through delegated OU administration or membership in Account Operators.

However, without access to their current hive, crafting a functional `NTUSER.MAN` is difficult. We advise against this approach.

2. Write to an existing profile share. Roaming profile shares with weak permissions allow direct placement of `NTUSER.MAN` following the earlier workflow. The target user's next logon loads your hive instead of their existing profile.

The roaming profile case has an additional consideration. Microsoft's documentation notes that when the profile server is unavailable, Windows falls back to a locally cached copy if one exists. A target who normally works on-premise but occasionally connects remotely may load your hive only when on the corporate network.

The only indicator for local persistence is the file write. Registry telemetry shows nothing because the registry APIs were not called.

Monitor for:

- File writes to `NTUSER.MAN` in any profile directory
- File writes to `NTUSER.MAN` on roaming profile shares
- Existence of `.MAN` files outside expected mandatory profile deployments
- Modifications to the `profilePath` attribute on AD user objects
- Hive load events correlating with unexpected `.MAN` files

Mandatory profiles are rare in modern environments. Outside of kiosk or shared workstation configurations, their presence warrants investigation.

Credit to Windows security researcher Jonas L for highlighting this "intended functionality".

Related Articles

Modern Adversary TTPs: The Rise of 'Read Teaming'Threat ResearchThreat Intelligence in Cyber Deception: A Planning GuideThreat ResearchUnderstanding Your Adversary: The Human Side of Threat IntelligenceThreat Research