

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Download More RAM: Dismantling Windows Operating System Defences with Mischievous Memory

Sam Collins, Tom Chothia, William Burgess, and Marius Muench, *The University of Birmingham;* David Oswald, *Durham University*

https://www.usenix.org/conference/usenixsecurity26/presentation/collins

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: Download More RAM: Dismantling Windows Operating System Defences with Mischievous Memory

Sam Collins
University of Birmingham

Tom Chothia
University of Birmingham

William Burgess
University of Birmingham

Marius Muench
University of Birmingham

David Oswald
Durham University

A Artifact Appendix

A.1 Abstract

Download More RAM describes an attack methodoly to subvert Windows defenses by introducing aliased memory from software, assuming the presence of DIMMs without SPD write protection. The artifact in this document describes how to reproduce the attack, including (i) checking whether installed DIMMs are susceptible to the attack, (ii) introducing and reading out aliased memory, (iii) patching Windows code integrity to enable loading of previously block-listed vulnerable drivers, and (iv) disabling AV and EDR software.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The attack interferes with the operation and security of the Windows operating system. Thus, data on the system may be at risk; unsuccessful runs of the attack may corrupt the system. We do not recommend to try out the attack on a live system with user data; all our experiments have been carried out on a test machine with a fresh Windows installation.

A.2.2 How to access

The development repository for *Download More RAM* is at: https://github.com/SamCollins1327/Download-More-RAM.

We also provide a stable version of the artifact at: https://doi.org/10.5281/zenodo.20331553.

A.2.3 Hardware dependencies

The full *Download More RAM* attack requires DDR4/DDR5 DIMMs without SPD write protection, being installed on a Secure Boot enabled platform.

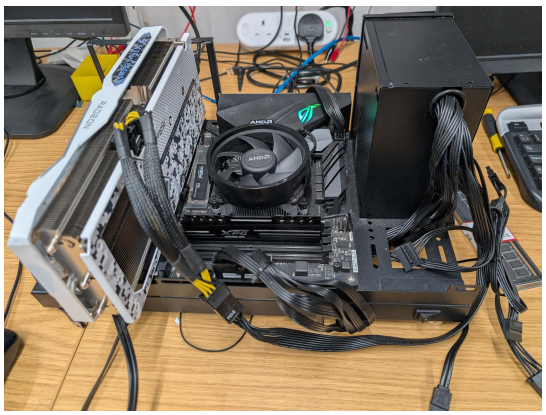


Figure 1: The experiment platform used for developing *Download More RAM*.

Known vulnerable setup. We developed *Download More RAM* on a ROG Strix B450F motherboard, with a Ryzen 7 2700 CPU, and a single 8GB stick of [Corsair Vengence, ADATA XPG, G.Skill Aegis] DDR4 memory.

Our setup is shown in Figure 1. Other hardware setups may require minor adjustments to the attack. For the artifact evaluation, we will provide assistance for getting the attack running on other hardware platforms.

A.2.4 Software dependencies

Our attack targets Windows 11, with the artifact specifically targeting build 28000.1. All tooling used is available in our artifact repository with the exception of Windows 11, and Primo Ramdisk. Windows 11 is free to use without a license and Primo Ramdisk offers a 30 day trial version.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Clone (or download) the artifact repository:

```
git clone git@github.com:SamCollins1327/Download-More-RAM.git
```

A.3.2 Build Instructions

We provide pre-compiled tools for ease of use. To build tools from source follow these steps.

- For *DMR_SPD_Tool*, *DMR_SPD_Tool_Headless*, *RTCore-write*, *RTCore_dump*, *RTCore_read*, *TrueSightKiller*, *TrueSightKiller_NoDriverLoad*, and *WinPmem-custom*: These tools are provided as visual studio projects. They can be built using Visual Studio Community Edition. For each project, open the corresponding .sln file, navigate to the build menu and build as release.
- For *helper_optimised.c*, *patcher.c*, and *sig_finder.c*: These tools can be compiled using gcc with no additional options with Mingw-w64.

A.3.3 Basic Test

On the system, run the `DMR_SPD_Tool` executable from an elevated command prompt. Select the DIMM slot corresponding to your installed DIMM and then select the option to check if SPD write protection is enabled (*"2. Check write protection"*). If the result indicates that SPD write protection is not enabled the the system is vulnerable to the Downlaod More RAM attack, assuming a pre-mitigation version of Windows 11 is running (i.e., any version before 26200.8246 and 26100.8246). If the result indicates that SPD write protection is enabled, then the DIMM SPD cannot be written from software and the system can be considered safe.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): A range of consumer DIMMs do not utilise write protection on their SPD chips, allowing to launch *Download More RAM* attacks from software. Experiment *(E1)* provides a custom testing utility for a DIMM's write protection status. We present our testing results in Table 2 of the paper.

(C2): *Download More RAM* allows introducing memory aliasing from a privileged user on Windows. Experiment *(E2)* replicates aliasing on a machine with DIMMs without SPD write protection and reads out contents of the aliased DIMM. The methodology is described in Section 6.1 and 6.2 of the paper.

(C3): Aliased memory can be re-written, ignoring privilege boundaries and memory protections. Experiment *(E3)* demonstrates the write primitive, on the example of patching the Windows code integrity module *skci.dll* in memory. This is described in Section 6.3 and 6.4 of the paper.

(C4): After disabling code integrity checks, *Download More RAM* allows to load previously blocked drivers, enabling further attacks. We demonstrate this in *(E4)* on the example of running TrueSightKiller, a tool disabling AV and EDR software. This corresponds to case study *"Disabling PPL protected AV/EDR"*, described in Section 7.2 of the paper.

A.4.2 Experiments

(E1): *[Verifying SPD Vulnerability] [30 human-minutes + negligible compute time + 1GB disk]:*
Preparation: On the target system running Windows 11, with secure boot and core isolation enabled (ideally with motherboard or equivalent chipset as described in A.2.3) install candidate DIMM. Have the precompiled `DMR_SPD_Tool` executable on disk ready.
Execution: Run the `DMR_SPD_Tool` executable from an elevated command prompt. From here select the DIMM slot corresponding to your installed DIMM, then select the option to check if SPD write protection is enabled (*"2. Check write protection"*).
Results: If the DIMM SPD is not write protected the SPD tool will print the message "NOT write protected". This means the DIMM is vulnerable to software induced aliasing. Conversely if the DIMM SPD is write protected the message will include "WRITE PROTECTED" and can be considered safe.
Note: The `DMR_SPD_Tool` was built on/for our target system and we have not verified it on other chipsets. Therefore, if the hardware is not avaliable then the Thaiphoon Burner v174 utility can be used for this step on a more universal set of systems, but not for SPD writing as is required in E2 without the paid license. SPD write protection can be checked via Thaiphoon Burner's *EEPROM->Data Protection Check* menu.

(E2): *[Alias Memory] [30 human-minutes + negligible compute time + 1GB disk]:*
Preparation: On the target system running Windows 11 (ideally with motherboard or equivalent chipset as described in A.2.3) install a DIMM verified to have no SPD write protection (e.g., the 8GB DDR4 DIMMs by Corsair Vengence, ADATA XPG, or G.Skill Aegis). Have the precompiled `DMR_SPD_Tool` executable on disk ready.
Execution: Run the `DMR_SPD_Tool` executable from an elevated command prompt. Select the DIMM slot corresponding to your installed DIMM, then select the option to alias the selected DIMM (*"3. Alias DIMM"*).

From an elevated command prompt run `bcdedit /set current removememory 8192` (assuming an 8GB DIMM) to prevent Windows from using the newly aliased memory. Reboot the system. This action is reversible via option (*"4. Un-alias DIMM"*).

Results: Upon reboot the system should report double the installed memory as is physically present i.e. 8GB->16GB. This can be verified both in the BIOS and in the msinfo utility in Windows.

(E3 & E4): *[Patching skci and killing Windows Defender] [30 human-minutes + negligible compute time + 18GB disk]:*

Preparation: On the target system, set up with aliased memory as described in E1 and E2, install Primo Ramdisk utility trial version and add the path to rxprd.exe in the installation folder to the PATH environmental variable.

Execution: Navigate into the Precompiled_Tools directory. (3) Set the path values in runme.bat to the directory of the precompiled tools file. Run runme.bat. In the case the signatures are not found (a) ensure invisible memory can be enabled in Primo Ramdisk (a restart may be required if not) and (b) verify manually the location of skci.dll in the current_decoded.raw file output using one function fingerprint from the patcher.c source code. For example if the module is found in the 30th-31st MB of the system, set the Ramdisk allocation in runme.bat line 12 to 28 (as is set by default).

Results: All functions should appear with a trailing *"...OK"* in the command output, signaling success. runme.bat will attempt to run truesightkiller.exe before and after patching skci.dll. The first attempt should give a warning that the driver cannot be loaded and deliberatley fail, the second (after the patch) should disable Windows defender by terminating its process.

A.5 Notes on Reusability

Our attack was implemented on the target machine described in Section A.2.3. Other machine setups may require modification of the attack, e.g., due to different memory scrambler implementations or BIOS-level defenses against the attack.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.