Independent Tests of
Anti-Virus Software

OS Credential Dumping 2022

LSASS Memory Test

TEST PERIOD: MAY 2022

LAST REVISION: 7TH SEPTEMBER 2022

COMMISSIONED BY: MICROSOFT

WWW.AV-COMPARATIVES.ORG

Introduction

This report was commissioned by Microsoft. The 2022 "OS Credential Dumping: LSASS Memory Test" provides an unbiased picture of a product's true current prevention and/or detection capabilities regarding LSASS credential dumping – in this specific case, of Microsoft Defender for Endpoint[1]. With regard to detection, one of the aims was to identify the correct Techniques and Sub-Techniques of the MITRE ATT&CK® framework. The results of the test will allow Microsoft to make any necessary product improvements. Whilst tests of this type are normally not intended for publication, Microsoft asked us to make a short summary of the report public. Microsoft was tested using its default settings.

This test looked only at specific aspects of the product. In order to achieve this, certain LSASS-specific hardening measures were not enabled. If the product had been tested with these protection features enabled, all 15 LSASS dumping attempts used in the test would have been blocked, but detailed detection information on the LSASS attack might not have been seen in the admin console.

Most of AV-Comparatives' tests cover the protection provided by the tested products. In this case, the aim was not to measure protection, but rather the detection information provided by the product, to be used in understanding targeted attacks. Hence, this report is of minimal interest to readers who only want to know about protection, and do not need to know about detection details.

With this type of advanced test, we provide vendors (the intended audience) with extensive relevant technical data about the test and results. As our public reports need to be concise and easy to read, we do not include all of this data in them. Microsoft received (as part of their internal report) in-depth details about how the attacks were done, including screen recordings, process monitor logs, PCAPs, etc.

About the LSASS process

The methods used by hackers in advanced persistent threats (APTs) can vary greatly from group to group. However, sooner or later in any attack, it is very likely that an attacker will attempt to access the LSASS process on an already compromised Windows host. The LSASS process is one of the most interesting Windows processes for an attacker, since it stores e.g. the Windows login data of the logged-in user, depending on the Windows configuration in plain text or in hash format. A possible scenario could be that on an already compromised host, further user sessions of useful domain users (Domain Admin, CEO etc.) or local users (Local Admin) are open. If an attacker has already compromised a privileged user account such as a Local Admin, or an unprivileged user account which (due to a misconfiguration) has debug privileges on the host, they can access the address memory of the LSASS process by the MITRE ATT&CK® Technique T1003.001[2] "OS-Credential Dumping: LSASS Memory".

Due to the high value and sensitivity of the LSASS process, it should be a top priority for a security product to detect malicious attacks on the LSASS process, and ideally block these and provide further detailed information about the attack, using the ATT&CK framework. Due to the increasing complexity of attacks on the LSASS process, this task is becoming more and more difficult for security vendors, and can be seen as be a quality feature for companies when evaluating a security product. An illustration of why security products need to protect against unauthorized LSASS accesses is shown by the RedCanary threat detection report[3] 2021, where OS Credential Dumping is ranked on place 5.

[1] Plan 2; Previously known as „Microsoft Defender Advanced Threat Protection (ATP)"
[2] https://attack.mitre.org/techniques/T1003/001/
[3] https://redcanary.com/threat-detection-report/techniques/lsass-memory/

Methodology

It should be noted that the LSASS Credential Dumping Test only tests one specific protection aspect (in contrast to e.g. AV-Comparatives' EPR[4] and ATP[5] Tests, which cover the entire attack chain). For the LSASS Credential Dumping Test, we used a fully patched Windows 10 host. The tester logged on to Windows as a minimal user (Windows shell starting in medium integrity), and then executed the respective LSASS dump POC ("proof of concept", i.e. custom malware), and the C2 Trojans in the black-box test, as a privileged user (high or system integrity). Since the focus of this test was not on the prevention or detection of local privilege escalation, it was taken that the tester already knew the credentials of the privileged user (Local Admin). We then looked at when the security product detected and/or prevented unauthorized access to the LSASS process, or declared the access to be unauthorized. We varied the use of the following factors in the LSASS Credential Dumping Test: *Credential Dumping Tools, Integrity Level, Living-off-the-Land Binaries, WIN32 APIs vs. Direct System Calls*, and *PPID Spoofing*.

Further details of the test methodology are given below:

- 15 different proof-of-concept malware samples (POCs) with different technical approaches were used to conduct the LSASS credential dumping test. These consisted of:
    - 8 POCs based on the white-box principle, with access to the Windows GUI. The POCs were executed from disk or in-memory (e.g. PowerShell).
    - 7 POCs based on the black-box principle, which included in-memory execution over a C2 channel.
- For the tests (white box and black box), the tester logged into Windows with an unprivileged user account (medium integrity), after which the respective POC was executed as a privileged user (high or system integrity).
- The Initial Access needed to open a stable C2 channel for the black-box in-memory test cases was performed by the tester, by executing the respective C2 malware sample with GUI access to the target host.
- The C2 malware sample is run as a privileged user (high or system integrity). This is because the test did not consider blocking malware or detecting local privilege escalation, but rather whether the tested products could detect unauthorised access to the LSASS memory.
- The malware sample used to open the C2 channel was not part of the evaluation process. If the C2 channel could not be opened at the beginning of the procedure (because the C2 malware sample had been prevented or detected with active response by the security product), the tester had to iterate the process to find a C2 malware sample which could be used to bypass the security product and open a stable C2 channel. This was because we wanted to evaluate whether the security prevented, or detected with active response, unauthorized access to the LSASS process, rather than the execution of the C2 malware sample (as in AV-Comparatives' APT Test).
- For each test case, we checked whether the security product provided prevention (actively prevented access to the address memory of LSASS) or detection with an active response (alert shown in the web console). One of the two criteria was sufficient for a rating as successful. No additional points were awarded for *both* prevention *and* detection with an active alert.
- A pure detection based on telemetry (without alert in web console), in combination with active threat hunting, was out of scope. No threat hunting was performed in this test, and so such cases were counted as misses.

[4] https://www.av-comparatives.org/enterprise/testmethod/endpoint-prevention-response-tests/
[5] https://www.av-comparatives.org/testmethod/advanced-threat-protection-tests/

- The tests were executed on a standard Windows 10 installation. Since we only wanted to measure the prevention and/or detection with active alert of the security product, all Windows LSASS-hardening measures were omitted in this test. This means that LSA Protection, Credential Guard, Restricted Admin Mode, etc. were not activated.

Setup and configuration

The following setup was used to perform the tests:

- Windows 10 host, default configuration *without any additional hardening measures like PPL, Credential Guard etc.* As this specific LSASS-Test focused on local OS-credential dumping of the LSASS memory, rather than on privilege elevation, lateral movement or similar tactics, an Active Directory environment was not required for the test.
- The product was configured by the manufacturer for the test, but once this configuration had been completed, it could not be further changed by either the manufacturer or the tester during the entire test. *The configuration had to be discussed with, and accepted by, the tester.* This was to ensure that there was no misconfiguration due to misunderstandings between the manufacturer and the tester, thus preventing invalid test results.
- The product had to be configured so that prevention was active.
- Microsoft was tested using its default settings.

Scope

- The results of the test focus on the prevention and detection (active response) capabilities in the case of an attacker trying to access the address memory of the LSASS process and steal credentials.
- There was no requirement for this enterprise product to use its default configuration. The manufacturer was allowed to configure the product with a more aggressive, harder configuration policy before the start of the tests, *as long as this configuration did not prevent general access to the address memory of the LSASS process*, e.g. in the form of Protected Process Light (PPL) or similar. The settings used had to be accessible via the product interface. Before the start of the tests, however, *the configuration had to be approved by the tester* in co-operation with the manufacturer. Microsoft was tested using its default settings.

Out of Scope

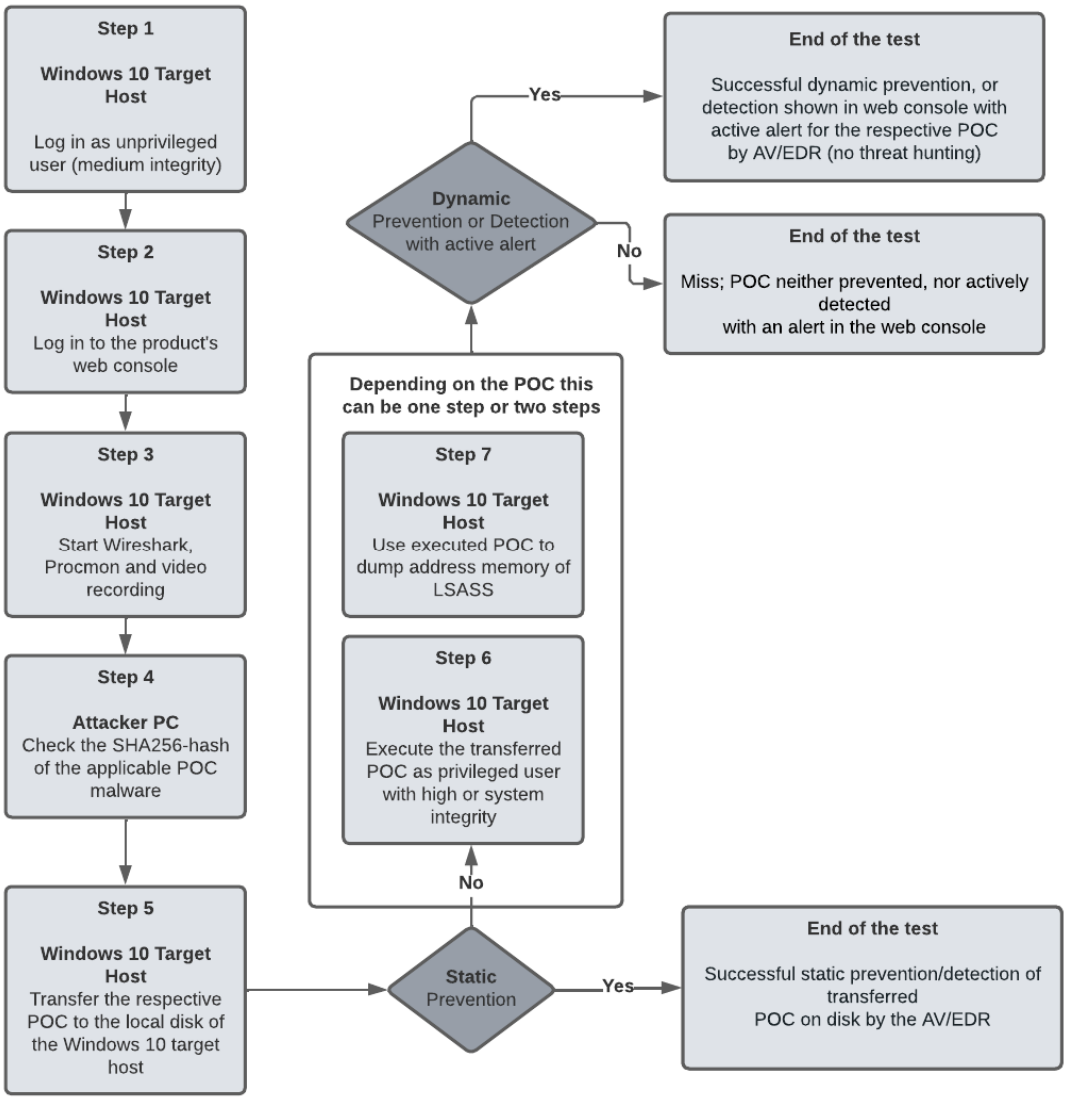
The following points were not evaluated in the test and were therefore out of scope:

- Evaluation of the escalation of privileges from an unprivileged user (medium integrity) to a privileged user (local admin, high integrity) or to the system account (system integrity).
- Evaluation of the prevention and detection capabilities with regard to the C2 malware samples used for the in-memory POCs to open a stable C2 channel in the first step. As already mentioned, the C2 malware sample used to open a stable C2 channel was tailored to the product being tested.
- Active threat hunting in web console.
- Hardening measures such as Protected Process Light (PPL), Credential Guard, Remote Credential Guard, Restrictive Admin Mode etc. were out of scope. This means that protection mechanisms like the ASR Rule "*Block credential stealing from the Windows local security authority subsystem*" that generally prevent access to the LSASS address memory, could not be activated by the vendor for the test. This was because we wanted to evaluate detection of unauthorized access to LSASS by the security product. Hence, we did not evaluate Windows hardening measures that protect the address space of LSASS in general (e.g. PPL), or that encrypt the stored credentials in LSASS (such as Credential Guard).
- Similar techniques in the product that try to protect unauthorized access to the address space of the LSASS process based on mechanisms like PPL. This is not about the overall technical capability of the tested product; rather, we wanted an unbiased measure of whether the execution of the respective POC is detected, and how the prevention/detection is handled by the product. For example, was only a general alert generated when accessing the address memory of LSASS, or was a more detailed detection based on TTPs of the ATT&CK framework produced?
- Decrypting an LSASS dump file that had been encrypted by the respective product. If the tester was able to dump the LSASS process with the respective POC without being prevented or detected with an active response in the web console, the vendor got a miss in that test case, even if the LSASS dump file was encrypted by the security solution.

Workflow

Because this test was about measuring the prevention and detection capability when dumping the LSASS with various different POCs, the way the privileges were escalated from an unprivileged user to a privileged user (local admin) was out of scope; it was taken that the tester already knew the credentials for the local admin account.

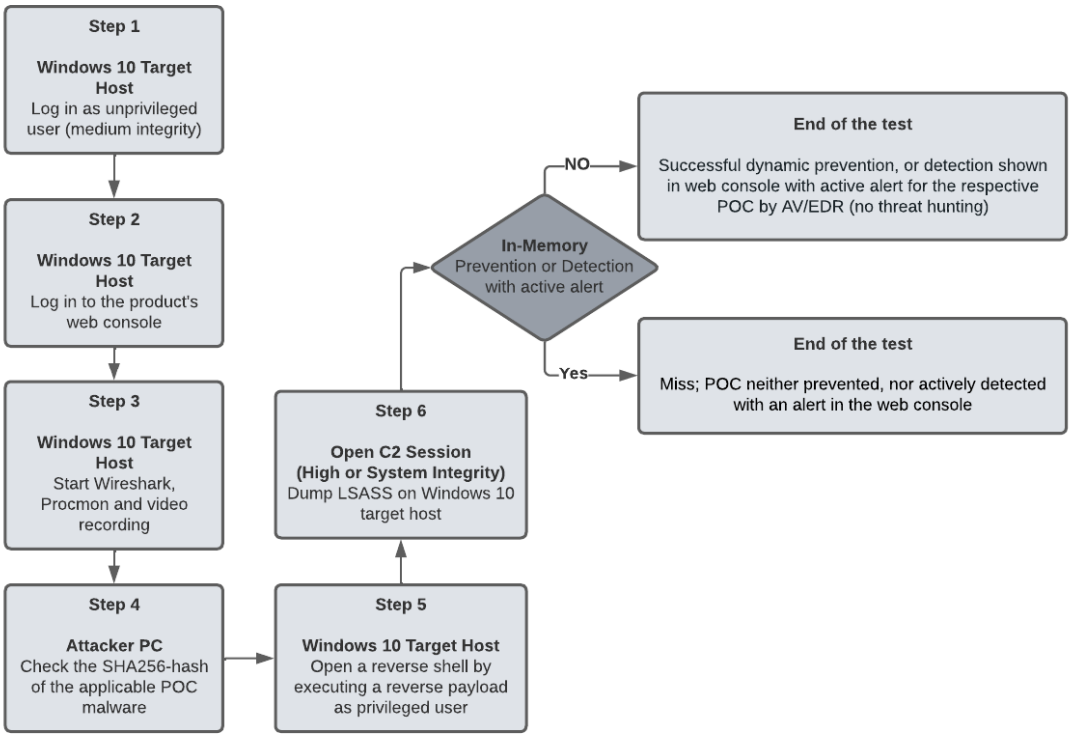
For the execution of the test cases via **white-box scenario,** the workflow was as follows:



Regarding Step 6: depending on the test case, we might additionally escalate from high integrity to system integrity before executing the respective LSASS dump POC.

Regarding Step 7: depending on the respective POC, there might be an additional step (Step 7). For example, if we take the Mimikatz POC, we have 2 steps, because we have to execute Mimikatz in Step 6 and use Mimikatz in Step 7 to access the address space of LSASS with a separate action. However, for the other POCs, Steps 6 and 7 are combined into one step, because executing these POCs (Step 6) with the respective arguments also automatically accesses the address space of LSASS.

For the execution of the test cases via **black-box scenario** (in-memory Windows 10 local credential dumping), the workflow looks like this:



Regarding 6: depending on the test case, we might additionally escalate from high integrity to system integrity before executing the respective LSASS dump POC.

Results

In the table below, we define the abbreviations used for the results:

Quality of EPP prevention	**Abbreviation**
Static prevention by the AV module after the POC was copied to disk.	Static
Dynamic prevention by the AV module on or shortly after POC execution.	Dynamic
In-memory prevention by the AV module. For example, if we were able to bypass the static and dynamic prevention and start/load Mimikatz, but as soon as we tried to touch LSASS, we were prevented in-memory by the AV module.	In-memory
Quality of EDR detection	**Abbreviation**
General detection without specific indicators of the corresponding MITRE Tactic, Technique or Sub-Technique.	General
Detection (active alert in the web console) using the ATT&CK Credential Access Tactic (https://attack.MITRE.org/tactics/TA0006/) in the case of dumping the address space of the LSASS process.	Tactic
Detection (active alert in the web console) using ATT&CK Technique OS Credential Dumping (https://attack.MITRE.org/techniques/T1003/) in the case of dumping the address space of LSASS.	Technique
Detection (active alert in the web console) using Sub-Technique OS Credential Dumping: LSASS Memory (https://attack.MITRE.org/techniques/T1003/001/) in the case of dumping the address space of LSASS.	Sub-Technique

Below we provide an overview of the attack methods used for each test case:

Test Case	Type	Description
01	Whitebox	Mimikatz (Process Herpaderping)
02	Whitebox	Native APIs DLL
03	Whitebox	Silent Process Exit
04	Whitebox	Alternative API Snapshot Function
05	Whitebox	MalSecLogon
06	Whitebox	Dump LSASS
07	Whitebox	Duplicate Dump
08	Whitebox	PowerShell Mimikatz
09	Blackbox	Invoke Mimikatz (PoshC2)
10	Blackbox	SafetyDump
11	Blackbox	Snapshot (PoshC2 RunPE)
12	Blackbox	Unhook (Metasploit Framework)
13	Blackbox	Reflective DLL (Metasploit Framework)
14	Blackbox	Invoke Mimikatz (PowerShell Empire)
15	Blackbox	Invoke-PPL Dump (PowerShell Empire)

The table below provides detailed information on prevention/detection in each test case:

Test Case	LSASS dumping was possible?	Extracting credentials (offline) from respective minidump file was possible?	Prevention by AV module	Detection by EDR module
01	Yes	Not required[6]	No*	No
02	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: LsassDump)	Sub-Technique
03	Yes	Yes, offline with Mimikatz	No*	Sub-Technique
04	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: LsassDump)	Not necessary
05	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: LsassDump)	Not necessary
06	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: LsassDump)	Sub-Technique
07	No	N/A	Dynamic (AV: ShaDumpz)	Not necessary
08	No	N/A	In-memory (AV: possible AMSI tampering)	Not necessary
09	Yes	Not required[6]	No*	Technique
10	Yes	Yes	No*	Sub-Technique
11	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: Wovdnut)	Not necessary
12	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: Sensitive credential memory read)	Sub-Technique
13	Yes	No, dump file written to disk was quarantined by the AV module	In-memory (AV: Meterpreter post exploitation tool)	Not necessary
14	No	N/A	In-memory (AV: Shaloti)	Technique
15	No	N/A	In-memory (AV: SysdUpdate)	Not necessary

Interpretation of the results

It can be seen in the table above that Microsoft Defender for Endpoint (with default settings) was able to prevent (AV[7] module) or detect (EDR module) the attack on LSASS in all but one of the test cases. For the test cases 02, 04, 05, 06, 11, 12 and 13, even though it was possible to dump the LSASS process with the respective POC, it was not possible to download the minidump file to our attacker machine. This was because after the minidump file was written to disk, it was immediately recognized and quarantined by the AV module. Hence, we were not able to extract credentials from the minidump file by using Mimikatz (offline) on our attacker machine. Furthermore, we observed that for the test cases 02, 06, 12 and 14, even though the execution of the POC was prevented by the AV module, the EDR module was still able to provide the ATT&CK Technique or Sub-Technique correctly. This would allow the analyst to better understand the attempted attack. In almost every attack scenario, dumping credentials from the LSASS process took place, but as the results (at time of testing) show, it was very difficult to extract the credentials from the LSASS process without at least creating a detection by the EDR module.

*) Microsoft added improvements to its product based on the findings from this test. In a retest in August 2022, Microsoft successfully prevented also the test cases 01, 03, 09 and 10.

[6] Not required, because Mimikatz displays the content of the LSASS dump.

[7] Microsoft Defender Antivirus

Copyright and Disclaimer

This publication is Copyright © 2022 by AV-Comparatives®. Any use of the results, etc. in whole or in part, is ONLY permitted after the explicit written agreement of the management board of AV-Comparatives prior to any publication. AV-Comparatives and its testers cannot be held liable for any damage or loss, which might occur as result of, or in connection with, the use of the information provided in this paper. We take every possible care to ensure the correctness of the basic data, but a liability for the correctness of the test results cannot be taken by any representative of AV-Comparatives. We do not give any guarantee of the correctness, completeness, or suitability for a specific purpose of any of the information/content provided at any given time. No one else involved in creating, producing or delivering test results shall be liable for any indirect, special or consequential damage, or loss of profits, arising out of, or related to, the use or inability to use, the services provided by the website, test documents or any related data.

For more information about AV-Comparatives and the testing methodologies, please visit our website.

AV-Comparatives
(September 2022)