Team Hydra

BLOG

Bypassing Credential Guard

In ye old days, a [hacker, red teamer, penetration tester, motivated child] would compromise a host, use an exploit to elevate or laterally move, and then Mimikatz their way to glory (ok, maybe not *just* in the old days). This is becoming increasingly more complicated to achieve. Many new technologies have been implemented to prevent or restrict access to credentials on a compromised host. Last year, Adam Chester (@_xpn_) wrote an excellent blog post on memory patching wdigest.dll to enable UseLogonCredentials. This is done without dropping files to disk or changing registry keys. I will touch on this briefly, but I strongly recommend reading his post before reading through this.

After reading XPN's post, myself and a co-worker started exploring other possibilities for bypassing / disabling protections using memory patching. Windows Defender Credential Guard seemed like an excellent target for this type of attack.

TL/DR && (POC || GTFO)

Wdigest can be enabled on a system with Credential Guard by patching the values of g_fParameter_useLogonCredential and g_IsCredGuardEnabled in memory.

PoC located here.

> "Introduced in Windows 10 Enterprise and Windows Server 2016, Windows Defender Credential Guard uses virtualization-based security to isolate secrets so that only privileged system software can access them. Unauthorized access to these secrets can lead to credential theft attacks, such as Pass-the-Hash or Pass-The-Ticket. Windows

> Defender Credential Guard prevents these attacks by protecting NTLM password hashes, Kerberos Ticket Granting Tickets, and credentials stored by applications as domain credentials."
>
> https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard

Currently, the most common way to overcome Cred Guard is to register a new Security Support Provider (SSP) (Mimikatz memssp or a custom one). This is generally loaded into memory and then captures and writes any credentials entered. This can be modified (seriously, @_xpn_ is a beast) to improve opsec but we wanted to see if there was a simpler and hopefully less identifiable way to do this.

The first thing we should check is if we can just enable UseLogonCredential on a system with Credential Guard enabled. Per MSDN:

> "When Windows Defender Credential Guard is enabled, neither Digest nor CredSSP have access to users' logon credentials. This implies no Single Sign-On use for these protocols."

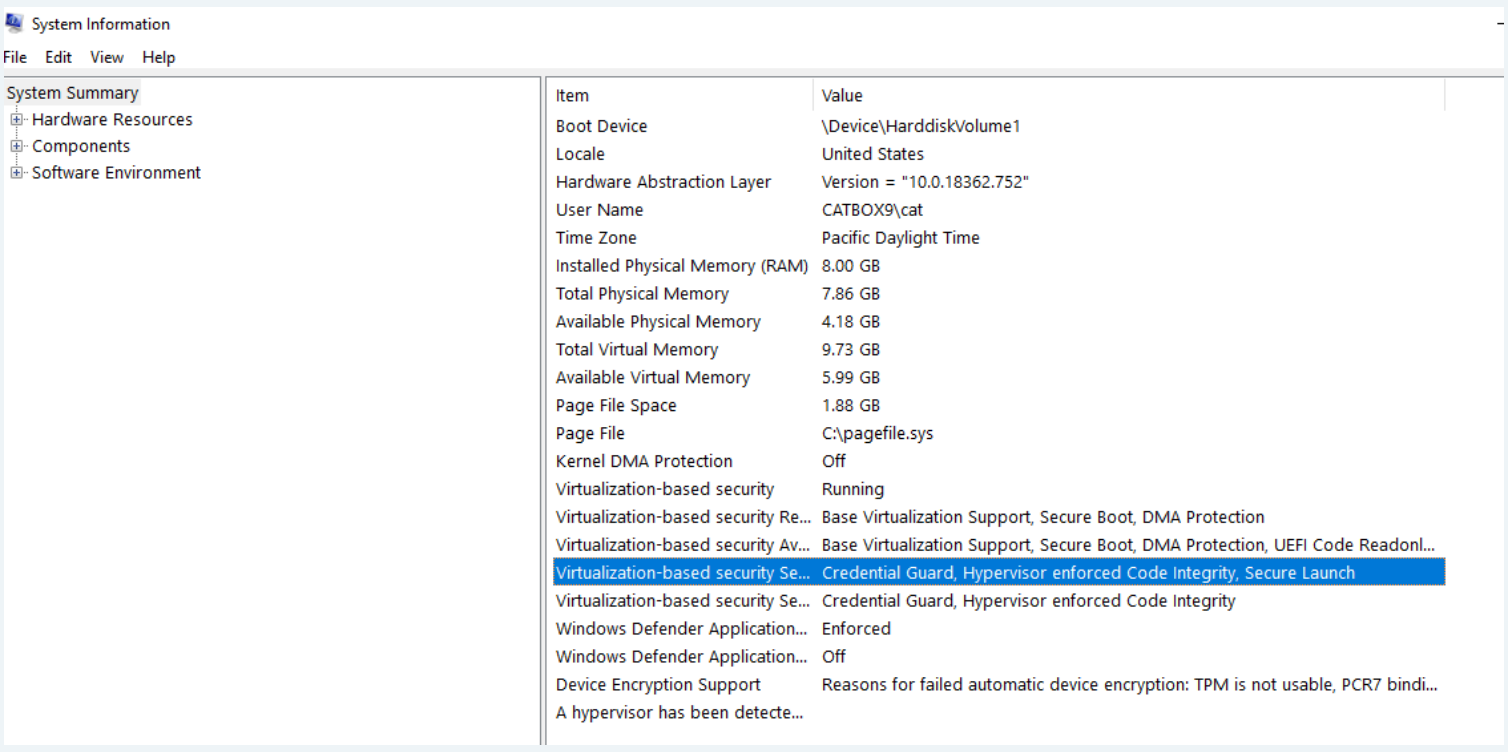
This *suggests* that enabling UseLogonCredential on a system with Credential Guard will not have any impact on an attacker's ability to get credentials from memory since Digest will not have access even if enabled. It still seems worth testing, just in case.

First, we identify the offset of g_fParameter_useLogonCredential. It should be noted that this will change between versions of wdigest.dll (but is static between systems with the same version). This can easily be done with any debugger.

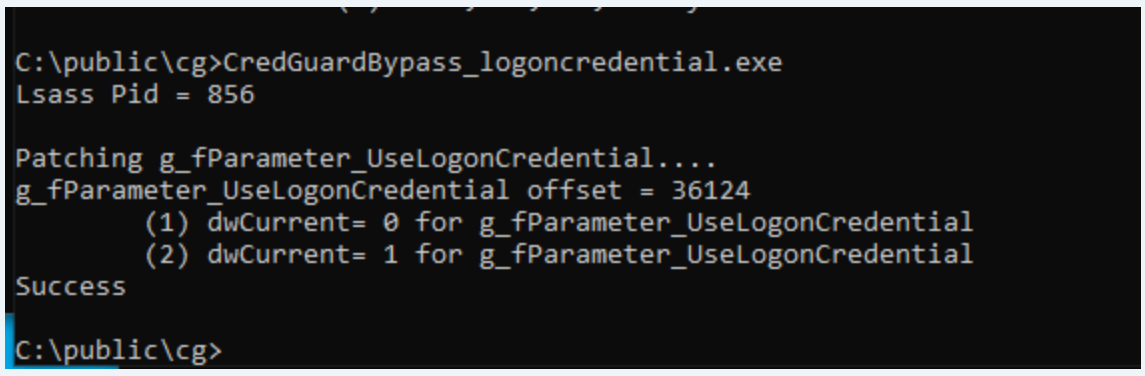
```
C:\Program Files (x86)\Windows Kits\10\Debuggers\x64+>cdb.exe -z C:\users\admin\desktop\wdigest.dll

Microsoft (R) Windows Debugger Version 10.0.10586.567 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\users\admin\desktop\wdigest.dll]

************** Symbol Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       symsrv*symsrv.dll*C:\Windows\Symbols*http://msdl.microsoft.com/download/symbols
Symbol search path is: symsrv*symsrv.dll*C:\Windows\Symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
ModLoad: 00000001`80000000 00000001`8003c000   C:\users\admin\desktop\wdigest.dll
wdigest!DllMainCRTStartupForGS:
00000001`80004ba0 4883ec28        sub     rsp,28h
0:000> x wdigest!g_fParameter_useLogonCredential
00000001`80036124 wdigest!g_fParameter_UseLogonCredential = <no type information>
0:000>
```

We can see wdigest!g_fParameter_useLogonCredential is 0x36124 bytes from the base of wdigest.dll (we are testing on Windows 10 Enterprise version 1909). With this information we can easily find and patch wdigest in memory on the host.

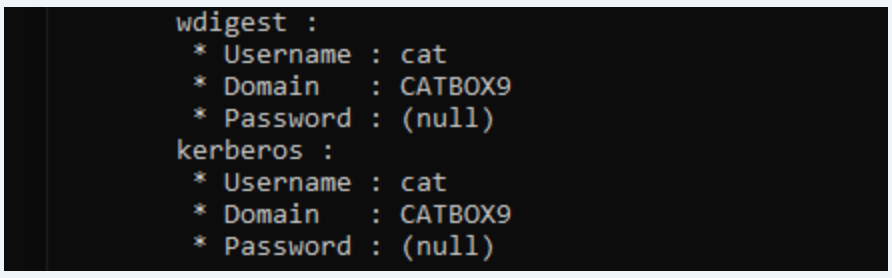
Let's boot up our system and ensure that Credential Guard is enabled.



Now, run our PoC that patches UseLogonCredential.



Finally, log in with a new user and see if we got credentials.....



Unsurprisingly, we are still unable to get new credentials. However, it seems like there may be more here to investigate, so we return to looking at wdigest.dll and see what other variables exist that could be of interest to us.

g_IsCredGuardEnabled is set to 1 when Cred Guard is enabled on a system. Unsetting this value seems to be worth trying.

Using the same technique as we used previously to get the offset to UseLogonCredential, we find the offset for g_IsCredGuardEnabled (here it is 0x35b88) and patch. Sadly, setting this value to 0 seemed to have no impact on anything relevant to our purposes. It turns out that the SpAcceptCredentials function in wdigest.dll checks both UseLogonCredential and IsCredGuardEnabled values to determine how to handle caching credentials. (We noticed later that XPN pointed out this value in his blog, but never goes farther into it).

So, what happens when we patch wdigest to enable UseLogonCredentials and unset IsCredGuardEnabled?

And now after we log in with a new session….

Clear text credentials! Note, this is NOT disabling Credential Guard but instead circumventing it by enabling wdigest.

As Credential Guard exists explicitly to help prevent elevated attackers from obtaining credentials from LSASS I reported this to Microsoft on principle.

Their response:

> "After investigating this issue, we do not believe this is a Credential Guard bypass. Credential Guard is meant to protect credentials that were cached while the feature is enabled. If a privileged user disables Credential Guard, then the feature cannot protect subsequent logons. We'll update our public documentation to clarify this behavior"

Given this response, I suspect this will be a reliable method of gaining clear text credentials on systems with Credential Guard enabled for the foreseeable future.

Memory patching host-based defenses has become a major aspect of modern red teaming. It is used to bypass AMSI, disable ETW, blind EDRs, and get cleartext credentials. This is just one more minor addition to the application of this type of technique. Hopefully posts like this will lead to increased visibility and better mitigation for this type of post-compromise action.

OPERATIONAL CONSIDERATIONS

The PoC provided IS NOT OPSEC SAFE. It is meant to demonstrate the concept. One of the biggest issues with this technique is that you are interacting directly with LSASS which is often looked for by EDRs (especially WriteProcessMemory). There have been many great posts about overcoming some of these hurdles (unhooking and using direct syscalls) but I leave it to you to pursue implementing this technique operationally.

DETECTION AND PREVENTION

Not much to say here that hasn't been said before. Monitoring LSASS, limiting administrators, network segmentation, all the usual suspects apply. A motivated and knowledgeable adversary that gains SYSTEM on a machine in your network will probably be able to accomplish what they need to on THAT system. The goal is to increase the cost in time, effort, and tooling to achieve that goal thus making your network less appealing as a target and increasing opportunities for detection and response.

N4kedTurtle

August 25, 2020

Uncategorized

NEXT POST

Implementing Direct Syscalls Using Hell's Gate

8 thoughts on "Bypassing Credential Guard"

Pingback: Zero-day for Apple (Safari), tools and malware news – Vulners Blog

Pingback: Publications | Outflank

Pingback: Direct Syscalls in Beacon Object Files – TerabitWeb Blog

Pingback: WdToggle – A Beacon Object File (BOF) For Cobalt Strike Which Uses Direct System Calls To Enable WDigest Credential Caching - Latest Hacking News Today - HakTechs

Pingback: A Beacon Object File (BOF) For Cobalt Strike Which Uses Direct System Calls To Enable WDigest Credential Caching | e-Shielder Security News

Pingback: WdToggle – A Beacon Object File (BOF) For Cobalt Strike Which Uses Direct System Calls To Enable WDigest Credential Caching – Parsec – Hacks

Pingback: Pentest Windows Active Directory – Mahyar Notes

Pingback: Deep into Windows Active Directory for Penetesters ! – Mahyar Notes

Leave a comment