AppLocker bypass by hash caching misuse

Ogtworek.github.io/PSBits/applockercachebypass.html

If you need to know more about AppLocker - <u>https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/applocker-overview</u>

As we know, AppLocker may use 3 different types of rules for whitelisting executable files: **Path**, **Publisher**, and **Hash**. The following whitepaper covers Hash rules. Such rules can be created through the wizard displayed in *secpol.msc* -> *Application Control Policies* -> *AppLocker* -> *Executable Rules* -> *Create New Rule...*

The wizard calculates a hash of the file, and then stores it within AppLocker policy. GUI will not display the hash value, but it may be checked from the PowerShell by issuing a command (Get-AppLockerPolicy -Local).ToXML()



Despite naming the hash "SHA256", the value **IS NOT** the SHA256 of the file. Instead, the custom way of calculating the hash is used, and Microsoft does not officially publish the method of calculating it. At the first use of the executable file, its hash is calculated again (using the same algorithm) and if hashes match, the AppLocker rule is applied.

As the hash calculation is expensive in performance terms (regardless the algorithm), Microsoft decided to cache the calculated hash, and store it in a very special way together with the corresponding file. Such cache should meet couple of requirements:

- tightly bound with its file met by using NTFS Extended Attributes (EA), <u>as</u> <u>documented by Microsoft</u>,
- protected from tampering/manipulation met by naming EA with **\$kernel** prefix. Such EAs can be modified/created only by Windows Kernel, which makes it protected from usermode processes,
- invalidated immediately if the executable file content changes met by adding purge prefix, <u>according to the documentation</u>.

For the AppLocker hash cache, the full name **\$kernel.purge.appid.hashinfo** is used. The complete list of EAs, including their contents may be displayed using the built-in command: **fsutil.exe file queryEA x:\path\file.exe**

```
Administrator: Command Prompt
C:\temp>fsutil file queryea test.exe
Extended Attributes (EA) information for file C:\temp\test.exe:
Total Ea Size: 0xa0
Ea Buffer Offset: 0
Ea Name: $KERNEL.PURGE.APPID.HASHINFO
Ea Value Length: 33
0000:
      00 00 00 41 49 44 31 00 00 00 00 00 00 00 00 20 ...AID1.....
0010: 00 00 00 5b 49 8b 1b 4f 19 c0 7d fc 9f 79 eb ca ...[I..0..}..y..
0020: f5 05 47 6e 80 a9 30 1d 82 06 65 27 97 dd 1a 41
                                                       ..Gn..0...e'...A
0030: a4 a0 b5
                                                       . . .
Ea Buffer Offset: 58
Ea Name: $KERNEL.PURGE.APPID.SIGNERINFO
Ea Value Length: 21
0000: 00 41 49 44 33 00 00 00 00 00 00 00 00 00 00 00 .AID3.....
0010: 00 00 00 00 00 00 00 00 00 fa b1 30 6f c8 30 d8
                                                       0020: 01
C:\temp>powershell -c Get-AppLockerFileInformation C:\temp\test.exe
Path
                       Publisher Hash
%OSDRIVE%\TEMP\TEST.EXE
                                 SHA256 0x5B498B1B4F19C07DFC9F79EBCAF505476E80A9301D820
C:\temp>
```

Effectively it means, such cache stays with a file, and when the file is manipulated, the old cache is deleted, and then re-calculated at the next run.

According to Microsoft, it's NTFS kernel driver responsible for the invalidation. If a malicious actor can change the file content without using the NTFS driver, it is possible to create a situation, where the cache does not reflect the real file being executed. It may be misused in the following scenario:

- 1. Administrator creates the AppLocker hash rule for a trusted file,
- 2. Malicious actor runs the file, to make sure its hash is calculated and then cached in the NTFS Extended Attribute,
- 3. Computer is turned off, and the hard disk content is manipulated, replacing the exe file with malicious code,
- 4. Computer is turned on, and the executable file may be run.

When AppLocker needs to determine if there is a hash rule allowing to run the file, it checks only the cache. As the cache remained intact, the file will be allowed to run, despite the changed content, and of course its real hash. The whole scenario can be seen on a <u>short video</u>, demonstrating it on a VM and its virtual drive.

			_		10 March 10	-
A loane line in the		Cod and Dep	and a	hand of the formation	The second secon	- Providence
and a second sec	-	OR/DECK/DAVIES	inspire-	ditte Manual	of Operanding.	SC-MARK
al incortant 21	diaman .	Services an emotion of	Table of the	-000 Teres	W Inside Factory States	Rendards.
	-	Conception Arrival and -	Televine .	100 Mar	and approximately and	NAME OF GROOM
	Parameter.	100000.0100.00	talata -	000 Mag	Textile	A DECEMBER OF
1. 2 test	-	the local with the local	Second Contraction in the	-	T the later line	C. Connell
- J Parlament 3.	Particulari -	1919 (min - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	Table online	dist lines	10. Avenue	122
	deleter.	100003110008	- Internetion	100 mm	Diate Lab	10000
	Rectal Controls	Lands Andrews	Cash of the	00 fee 00 he	100 million	ALC: YES
the second second	diama di seconda di se	2012/02/14/14/04	reprint	No. of Concession, Name	I de baselones.	1000
	distant of the local distant	100000000000000000000000000000000000000	Salaran .	dia here	analysi has been jun-	Concernance of
Chart B	distant and	2010/02/11/02/08	"lash-srive-	901 Hori		Contraction of the local division of the loc
the supervision in	Parallel .	00002-41100-00	Treat to the	012 Test		Concert States
	distant and	providence in traditional Constant and tablease	Data John	and then	12. Arturn	- Ministerio
	Transa -	TERMINAL PROPERTY.	Tank to be a construction of the second seco		all resp	100000
L These P		CONTRACTOR OF CONTRACT,	and the second se	100.000	American Sector	ALC: No. of Concession, Name
Contractor 3	d'united	33/80141-644	and the second s		A Darmann	and the second of
C and and a	+ FEL Jasmurier				· · · · · · · · · · · · · · · · · · ·	60 - MC-
in Application 1					in the trans-	 SAUGRAS
and the second s		1.0			and the investment of	1003057
- Republic		101110 he was (1.1.1) - 1014			do federate	CALLER &
C Aprilianys					ill me	 MACHINE
in the Party of Longer						22-24
of several second se	arms in	And the state of the second state of the secon				A Course
			and the second se			Darbourg
of alloyed	haird mil	him hat course in				1000000
- K Buller	18 - Carl	man horse				10000
		Destaurant Destaura 10	interaction (1005500
	and the					PROVIDE
C heterarth	and because he	d Lin Language				C (400)
- C Antonette						

Even if the cache misuse can be see as a weakness, it does not open any new way of attacking the protected Windows OS. If a malicious actor can manipulate the file content without using NTFS driver, it may compromise Windows security in many other ways too.