Defence Evasion Technique: Timestomping Detection – NTFS Forensics

By inversecos

Published: 2022-04-28 · Archived: 2026-04-06 00:29:39 UTC

USN JOURNAL

Offset	FileName	USN	Timestamp	Reason	MFTReferen	MFTReferenceSeqNo	MFTParentR	MFTParentR	FileAttribute	MajorVersio	MinorVersio	SourceInfo	SecurityId
0x519ECEB0	file.txt	1369362096	2022-04-21T02:45:38Z	FILE_CREATE	91408	10	28878	9	archive	2	0	0x00000000	0
0x519ECF00	file.txt	1369362176	2022-04-21T02:45:38Z	CLOSE+FILE_CREATE	91408	10	28878	9	archive	2	0	0x00000000	0
0x519ECF50	file.txt	1369362256	2022-04-21T02:45:38Z	CLOSE+FILE_DELETE	91408	10	28878	9	archive	2	0	0x00000000	0
0x519EDA40	file.txt	1369365056	2022-04-21T02:45:38Z	FILE_CREATE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDA90	file.txt	1369365136	2022-04-21T02:45:38Z	DATA_EXTEND+FILE_CREATE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDAE0	file.txt	1369365216	2022-04-21T02:45:38Z	CLOSE+DATA_EXTEND+FILE_CREATE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDB30	file.txt	1369365296	2022-04-21T02:45:38Z	OBJECT_ID_CHANGE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDB80	file.txt	1369365376	2022-04-21T02:45:38Z	CLOSE+OBJECT_ID_CHANGE	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A028C8	file.txt	1369450696	2022-04-21T02:53:57Z	BASIC_INFO_CHANGE	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A02918	file.txt	1369450776	2022-04-21T02:55:01Z	BASIC_INFO_CHANGE+CLOSE	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A04138	file.txt	1369456952	2022-04-21T02:55:01Z	RENAME_OLD_NAME	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A04188	file.txt	1369457032	2022-04-21T02:55:01Z	RENAME_NEW_NAME	91701	5	23220	7	archive	2	0	0x00000000	0
0x51A041D8	file.txt	1369457112	2022-04-21T02:55:01Z	CLOSE+RENAME_NEW_NAME	91701	5	23220	7	archive	2	0	0x00000000	0
0x51A04228	file.txt	1369457192	2022-04-21T02:55:01Z	SECURITY_CHANGE	91701	5	23220	7	archive	2	0	0x00000000	0
0x51A04278	file.txt	1369457272	2022-04-21T02:55:01Z	CLOSE+SECURITY_CHANGE	91701	5	23220	7	archive	2	0	0x00000000	0

Forensic analysts are often taught two methods for detecting file timestomping that can lead to blind spots in an investigation. The two most well-taught methods for analysts to detect timestomping are:

- Compare the $STANDARD_INFORMATION timestamps vs the $FILE_NAME timestamps in the Master File Table (MFT)
- Look for nanoseconds in a timestamp matching “0000000” as this often shows the use of an automated tool (i.e. Metasploit)

These two detection methods are based on two fallacies that I will explore in this blog post:

- **Myth 1:** $FILE_NAME timestamps cannot be timestomped
- **Myth 2:** Attacker tools cannot alter nanoseconds in a timestamp

INTRODUCTION TO TIMESTOMPING

Timestomping is a technique where the timestamps of a file are modified for defence evasion. Threat actors often perform this technique to blend malicious files with legitimate files so that when an analyst is performing IR, critical evidence escapes detection.

Timestomping using tools like Cobalt Strike (offensive-security tool), Timestomp.exe (timestomping tool) and Metasploit (offensive-security framework) will result in timestamp changes to the MAC(b) times in an MFT file’s $STANDARD_INFORMATION attribute. These MAC(b) times stand for:

- Modified
- Accessed
- Changed (MFT change)
- Birth (Creation time)

There are two attributes that record times in an MFT file – the $STANDARD_INFORMATION ($SI) and the $FILE_NAME ($FN) attribute. Each of these attributes stores the MAC(b) times for the file accordingly. For files

with filenames that are longer, there will be two corresponding $FN MAC(b) attributes totalling another 8 timestamps on top of the existing $SI timestamps.

Modification of the $SI timestamp is the most common method of timestomping as it can be modified at the user-level using a set of API calls. However, modification of the $FN attribute requires the kernel – OR abuse of how the $FN timestamps are set (when a file is renamed or moved).

As such, there are two methods for modifying the $FN timestamps:

- **Method 1**
  Modification of $FN on older operating systems where Patch Guard hasn't been introduced using Windows API calls to the native APIs NtSetInformationFile and NtQueryInformationFile.
- **Method 2**
  Modification of $FN on any OS by timestomping the $SI attribute and then moving or renaming the file to copy the $SI time into the $FN time. This technique was discussed in this Black Hat Presentation.

Threat Actors Utilising Timestomping

Just to demonstrate for those who haven't come across timestomping in an investigation – threat actors and malware often use this technique. Some notable threat actors that have used timestomping during their attacks include (and is not limited to):

- IRON TWILIGHT (APT28)
- IRON HEMLOCK (APT29) – Timestomped their backdoors
- TIN WOODLAWN (APT32) – Timestomped raw XML scheduled task files and modified the CREATE times.
- NICKEL GLADSTONE (APT38) – Modifying file times to match other files on the system.
- NICKEL ACADEMY (Lazarus) – Copying timestamp from calc.exe to their malicious dropped files (this is something that Cobalt Strike does)

If you're interested in reading more about this - check out the Mitre attack page for this technique.

ATTACK METHODOLOGY: Timestomping $FN

If a threat actor timestomps the $SI attribute, and then moves or renames the file – Windows will copy the timestomped $SI times into the $FN attributes. On older versions of Windows where Patch Guard does not exist, you can use SetMace to alter the $FN timestamps by relying on the API calls (NTSetInformationFile and NTQueryInformationFile).

Step 1: Timestomp the $SI attributes by setting nanosecond precision

The tool I am using here is nTimetools and as you can see here, this already sets the nanosecond to an arbitrary attacker-defined number. This defeats the first detection that's commonly taught to "look for .0000000 in the nanoseconds". This has been documented by Forensics Wiki, Mari DeGrazia and Harlan Carvey.

```
PS C:\Users\Lina Lau\Desktop\nTimetools-master> .\nTimestomp_v1.2_x64.exe -F "C:\Users\Lina Lau\Desktop\file.txt" -M "20
20-05-19 12:34:56.7890123" -A "2020-05-19 12:34:56.7890123" -C "2020-05-19 23:59:59.7890123" -B "2020-05-19 23:59:59.789
0123"
nTimestomp, Version 1.2
Copyright (C) 2019 Benjamin Lim
Available for free from https://limbenjamin.com/pages/ntimetools

Filesystem type:                NTFS
Filename:                       C:\Users\Lina Lau\Desktop\file.txt
File size:                      4

File timestamp successfully set

[M] Last Write Time:            2020-05-19 12:34:56.7890123 UTC
[A] Last Access Time:           2020-05-19 12:34:56.7890123 UTC
[C] Metadata Change Time:       2020-05-19 23:59:59.7890123 UTC
[B] Creation Time:              2020-05-19 23:59:59.7890123 UTC
```

When you analyse the MFT just to track the changes you can see that these are the timestamps for the $SI and $FN attributes.

$STANDARD_INFORMATION Timestamps:

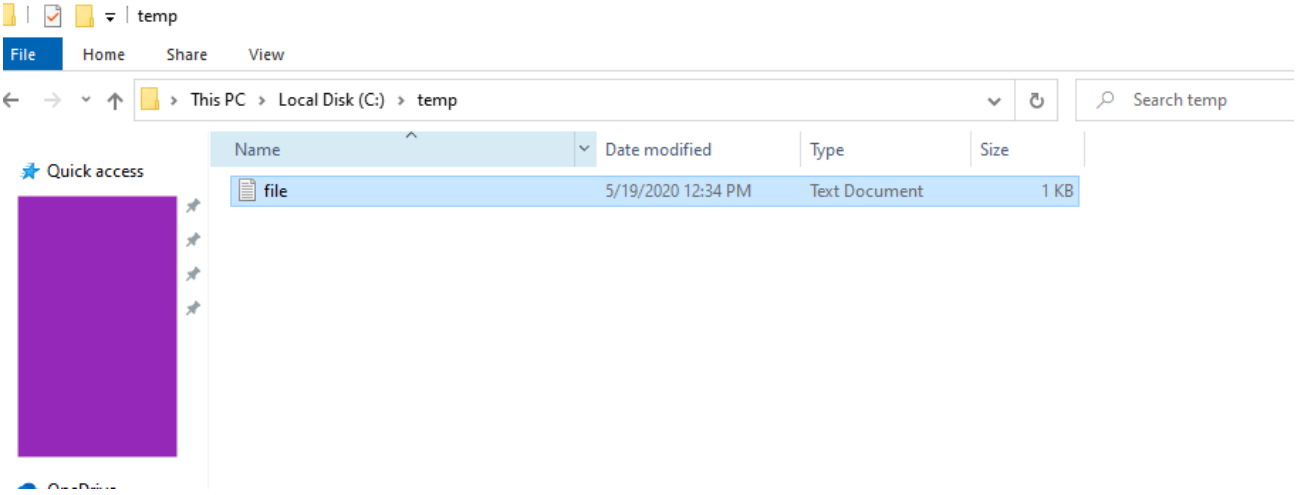
Creation – 2020-05-19 12:34:56
Modification - 2020-05-19 12:34:56
MFT change – 2020-05-19 12:34:56
Last Access - 2020-05-19 12:34:56

$FILENAME Timestamps:

Creation – 2022-04-21 02:45:38
Modification - 2022-04-21 02:45:38
MFT change - 2022-04-21 02:55:01
Last Access - 2022-04-21 02:45:38

Step 2: Move the file

In this instance I moved my “file.txt” from my Desktop into the Temp folder.



Step 3: Check the $FI and $SI times have been altered

When I dumped out the MFT and reviewed the timestamps for my file C:\temp\file.txt – these were the following timestamps:

$STANDARD_INFORMATION Timestamps:

Creation – 2020-05-19 12:34:56

Modification - 2020-05-19 12:34:56

MFT change – 2020-04-21 02:55:01

Last Access - 2020-05-19 12:34:56

$FILENAME Timestamps:

Creation – 2020-05-19 12:34:56

Modification - 2020-05-19 12:34:56

MFT change - 2020-05-19 12:34:56

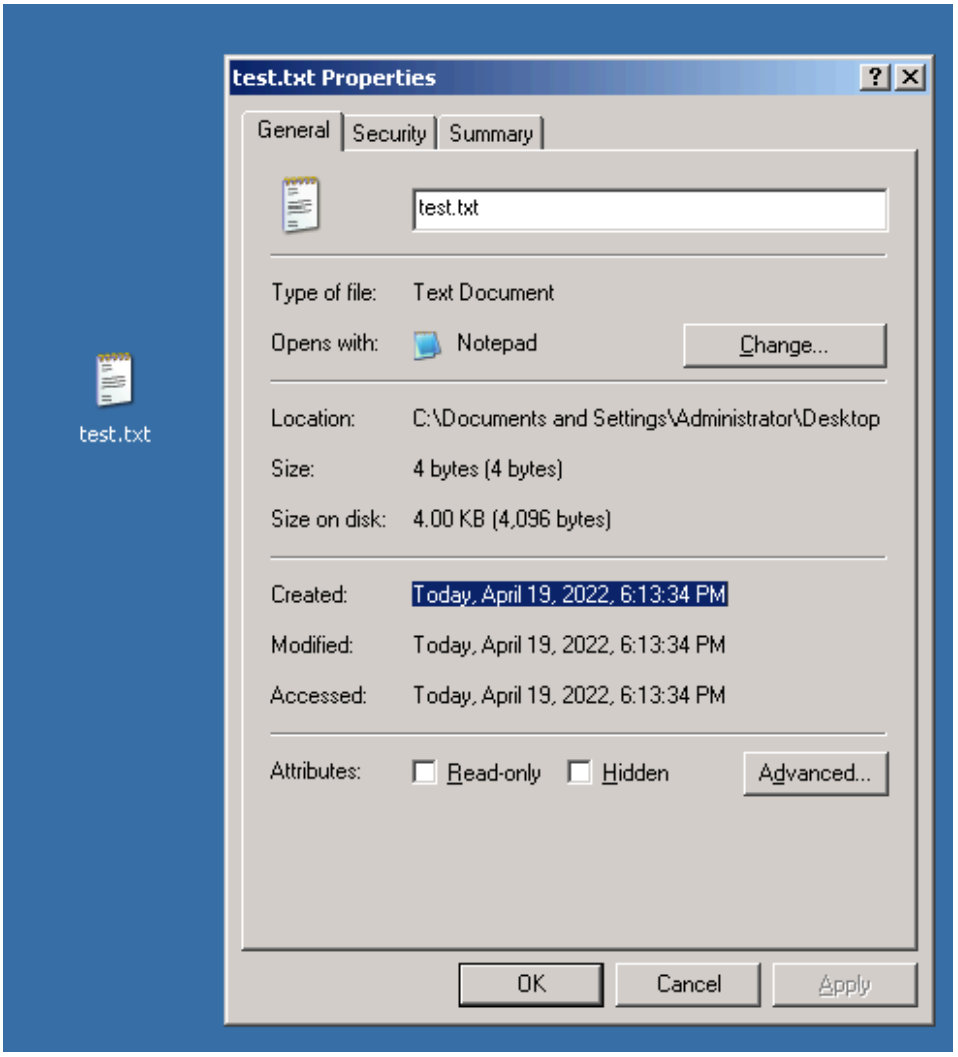
Last Access - 2020-05-19 12:34:56

As you can see here, the $FN attributes have been changed. The threat actors at this point just need to timestomp the MFT change time and then you have a perfect set of timestamps.

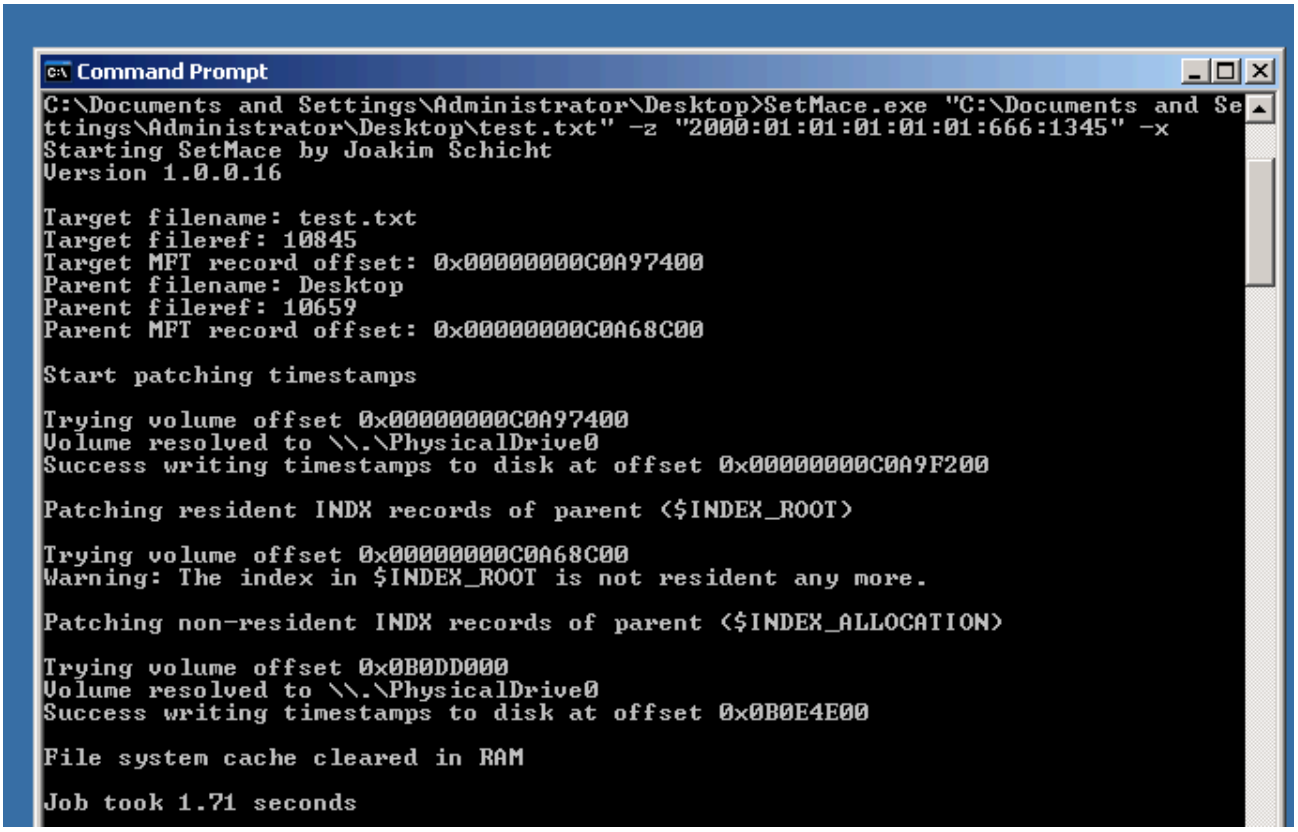
Step 4: Alternate Method

If you're faced with an older version of windows without Patch Guard – you can use the SetMace tool and rely on API manipulation of the timestamps. I performed this on a 32-bit Windows 2003 Server and got the same results:

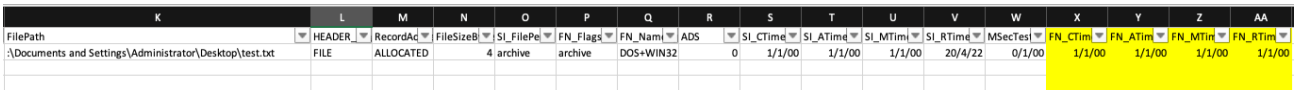
First I created a file on my Desktop named "test.txt"



Then proceeded to run SetMace to timestomp both $FI and $FN attributes:



And then I parsed out the $MFT and you can see here that the $FN and $SI attributes were both manipulated:



FilePath	HEADER	RecordAc	FileSizeB	SI_FilePe	FN_Flags	FN_Nami	ADS	SI_CTime	SI_ATime	SI_MTim	SI_RTime	MSecTes	FN_CTim	FN_ATim	FN_MTim	FN_RTim
:\Documents and Settings\Administrator\Desktop\test.txt	FILE	ALLOCATED	4	archive	archive	DOS+WIN32	0	1/1/00	1/1/00	1/1/00	20/4/22	0/1/00	1/1/00	1/1/00	1/1/00	1/1/00

DETECTION METHODOLOGY

For most cases of timestomping where an attacker uses one of the following methods below - the detection methods of comparing $FI and $SI along with looking at nanosecond precision will suffice.

- Cobalt Strike
- Metasploit
- Timestomp.exe
- SetMace.exe
- APIs to manipulate timestamps

However, as demonstrated above, it's almost trivial to bypass these two detection mechanisms which will force an analyst to consider other methods for detection. The best method I have found for detecting these anomalies is by analysing the USN Journal file and the $Logfile. On busy systems, the $Logfile may be overwritten very quickly which would force an analyst to consider pulling a $Logfile from a shadow copy. However, just the USN Journal file will show enough information for an analyst to "question" the authenticity of the timestamps.

USN Journal Detection

As you can see in my parsed USN Journal output below, there are multiple operations that are tracked:

- FILE_CREATE
- RENAME_OLD_NAME
- RENAME_NEW_NAME

For all these timestamps that are tracked, you can see the timestamps correlating to 21st April 2021 around 2:45. These timestamps greatly contradict the timestamps stored in the MFT where the timestomped times date back to 2020. By reviewing the USN Journal and seeing this anomaly, an analyst should have alarm bells ringing. In my opinion, this is a more foolproof way of detecting timestomping versus looking for nanosecond precision / comparing $FN to $SI.

USN JOURNAL													
Offset	FileName	USN	Timestamp	Reason	MFTReferen	MFTReferenceSeqNo	MFTParentR	MFTParentR	FileAttribute	MajorVersio	MinorVersio	SourceInfo	SecurityId
0x519ECEB0	file.txt	1369362096	2022-04-21T02:45:38Z	FILE_CREATE	91408	10	28878	9	archive	2	0	0x00000000	0
0x519ECF00	file.txt	1369362176	2022-04-21T02:45:38Z	CLOSE+FILE_CREATE	91408	10	28878	9	archive	2	0	0x00000000	0
0x519ECF50	file.txt	1369362256	2022-04-21T02:45:38Z	CLOSE+FILE_DELETE	91408	10	28878	9	archive	2	0	0x00000000	0
0x519EDA40	file.txt	1369365056	2022-04-21T02:45:38Z	FILE_CREATE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDA90	file.txt	1369365136	2022-04-21T02:45:38Z	DATA_EXTEND+FILE_CREATE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDAE0	file.txt	1369365216	2022-04-21T02:45:38Z	CLOSE+DATA_EXTEND+FILE_CREATE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDB30	file.txt	1369365296	2022-04-21T02:45:38Z	OBJECT_ID_CHANGE	91701	5	28878	9	archive	2	0	0x00000000	0
0x519EDB80	file.txt	1369365376	2022-04-21T02:45:38Z	CLOSE+OBJECT_ID_CHANGE	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A028C8	file.txt	1369450696	2022-04-21T02:53:57Z	BASIC_INFO_CHANGE	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A02918	file.txt	1369450776	2022-04-21T02:55:01Z	BASIC_INFO_CHANGE+CLOSE	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A04138	file.txt	1369456952	2022-04-21T02:55:01Z	RENAME_OLD_NAME	91701	5	28878	9	archive	2	0	0x00000000	0
0x51A04188	file.txt	1369457032	2022-04-21T02:55:01Z	RENAME_NEW_NAME	91701	5	23220	7	archive	2	0	0x00000000	0
0x51A041D8	file.txt	1369457112	2022-04-21T02:55:01Z	CLOSE+RENAME_NEW_NAME	91701	5	23220	7	archive	2	0	0x00000000	0
0x51A04228	file.txt	1369457192	2022-04-21T02:55:01Z	SECURITY_CHANGE	91701	5	23220	7	archive	2	0	0x00000000	0
0x51A04278	file.txt	1369457272	2022-04-21T02:55:01Z	CLOSE+SECURITY_CHANGE	91701	5	23220	7	archive	2	0	0x00000000	0

$Logfile Detection

It's not always feasible that the $Logfile will store the information that you're looking for – especially on busy disks. However, for the sake of showing the detection – I did this on my "not busy" VM.

What you want to look for the $Logfile are two logs pertaining to:

- Operation: CreateAttribute
- Filename: file.txt (or your filename)
- CurrentAttribute: $FILE_NAME

When performing timestomping where you're trying to overwrite the $FN attribute – there are two logs of interest in the $Logfile:

- The original $FILE_NAME timestamp when the file was originally created
- The new $FILE_NAME timestamp when the files $FI attribute is timestomped

Therefore, when considering the detection, you should look for both of these lines in the $Logfile. This is incredibly useful to see and is also another great method for determining if timestomping has occurred – especially when considering that the time will not match the time in the $MFT $FI and $SI attributes.

Just to show you the example from my test – this is what the two contradicting log lines look

lf_Offset	lf_RedoOperation	lf_UndoOperation	lf_FileName	lf_CurrentAttribute	lf_FN_CTime	lf_FN_ATim	lf_FN_MTim	lf_FN_RTim	lf_FN_AllocS	lf_FN_RealS	lf_FN_Flags	lf_FN_Name
0x03C6DB30	CreateAttribute	DeleteAttribute	file.txt	$FILE_NAME	2022-04-21T05	2022-04-21T	2022-04-21T	2022-04-21T	0	0	archive	DOS+WIN32
0x03DB6C30	CreateAttribute	DeleteAttribute	file.txt	$FILE_NAME	2020-05-19T12	2020-05-19T	2020-05-19T	2020-05-19T	0	0	archive	DOS+WIN32

Happy hunting!

REFERENCES

https://github.com/limbenjamin/nTimetools

http://windowsir.blogspot.com/2014/07/file-system-ops-effects-on-mft-records.html

https://github.com/jschicht/SetMace

http://forensicinsight.org/wp-content/uploads/2013/06/F-INSIGHT-NTFS-Log-TrackerEnglish.pdf

https://az4n6.blogspot.com/2014/10/timestomp-mft-shenanigans.html

https://forensicswiki.xyz/wiki/index.php?title=Timestomp#:~:text=Timestomp%20is%20a%20utility%20co,timestamp%2Drelated%20information%20on%20files.

https://www.blackhat.com/presentations/bh-usa-05/bh-us-05-foster-liu-update.pdf

http://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FNT%20Objects%2FProcess%2FNtSetInformationProcess.html

---

Source: https://www.inversecos.com/2022/04/defence-evasion-technique-timestomping.html