Using dotnetfile to get a Sunburst timeline for intelligence gathering

r136a1.info/2022/06/18/using-dotnetfile-to-get-a-sunburst-timeline-for-intelligence-gathering/

Jun 18, 2022 • tool, malware

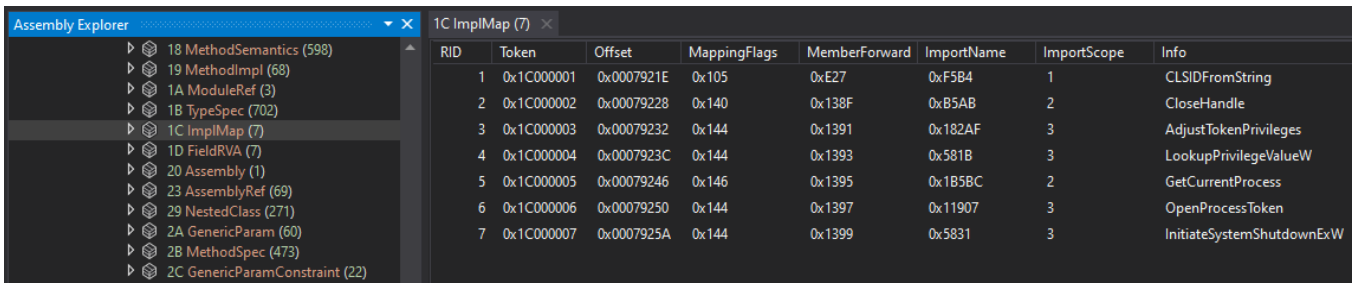
You may have heard of dotnetfile, a library to extract header information from .NET assemblies. Basically, these files are made of the common language runtime (`CLR`) data located in the .NET header and the actual byte code, both “encapsulated” in a PE file. Compared to the PE header of an unmanaged native file, the CLR header contains much more runtime information. Some of this data can be useful for static malware detection, threat hunting and intelligence gathering as I’ll show in this blog post.

We’ll take a look at a part of the SolarWinds incident, the malware known as Sunburst. More precisely, we’ll create a timeline of the legitimate and backdoored Orion IT management software DLLs named `SolarWinds.Orion.Core.BusinessLayer.dll` with the help of `dotnetfile` . You’ll see how easy it is to get an overview of the this part of the attack with just a few lines of code.

One of the characteristics of the `Sunburst` backdoor is the usage of various unmanaged functions. .NET offers a mechanism called P/Invoke that let’s you use unmanaged Windows API functions in your managed code. The information which functions are used can be found in the CLR metadata table named `ImplMap`. We can easily extract this data from a .NET assembly with the help of `dotnetfile` as we’ll see.

Create a Sunburst timeline based on unmanaged backdoor functions

As initially described, the developers of Sunburst used unmanaged functions in their backdoor code. This list of functions be seen with a CLR header viewer like the one built into dnSpy:



RID	Token	Offset	MappingFlags	MemberForward	ImportName	ImportScope	Info
1	0x1C000001	0x0007921E	0x105	0xE27	0xF5B4	1	CLSIDFromString
2	0x1C000002	0x00079228	0x140	0x138F	0xB5AB	2	CloseHandle
3	0x1C000003	0x00079232	0x144	0x1391	0x182AF	3	AdjustTokenPrivileges
4	0x1C000004	0x0007923C	0x144	0x1393	0x581B	3	LookupPrivilegeValueW
5	0x1C000005	0x00079246	0x146	0x1395	0x1B5BC	2	GetCurrentProcess
6	0x1C000006	0x00079250	0x144	0x1397	0x11907	3	OpenProcessToken
7	0x1C000007	0x0007925A	0x144	0x1399	0x5831	3	InitiateSystemShutdownExW

To extract the list of functions, we can use the following example script from the `dotnetfile` documentation:

```
from dotnetfile import DotNetPE

dotnet_file = DotNetPE('/Users/<username>/my_dotnet_assembly.exe')

if dotnet_file.metadata_table_exists('ImplMap'):
    unmanaged_functions = dotnet_file.ImplMap.get_unmanaged_functions()

    for unmanaged_function in unmanaged_functions:
        print(f'{unmanaged_function}')
```

Result:

```
CLSIDFromString
CloseHandle
AdjustTokenPrivileges
LookupPrivilegeValueW
GetCurrentProcess
OpenProcessToken
InitiateSystemShutdownExW
```

In comparison, the legitimate versions of this DLL do not use Windows API functions or only `CLSIDFromString` as introduced in Orion version `2015.1` . With this knowledge, we can create a script that runs on all copies of `SolarWinds.Orion.Core.BusinessLayer.dll` that we can get (e.g. from Virustotal) to show which versions have been backdoored. Additionally to the list of unmanaged functions retrieved via `ImplMap.get_unmanaged_functions`, we get the assembly version information of each file via `Assembly.get_assembly_version_information`. At last, we get the compilation timestamps with `pefile` which is easy as `dotnetfile` is build on top of it. The timestamps are used to sort the data in ascending order to see when the backdoor was added and removed.

Example script:

```
import os
import sys
import lief

from dotnetfile import DotNetPE
from datetime import datetime
from hashlib import sha256
from distutils.version import LooseVersion
from typing import List, Dict

lief.logging.disable()


def deduplicate(data_list: List[Dict]) -> List[Dict]:
    result = []

    for data in data_list:
        # Add file info if version not in the list
        if not any(i['Version'] == data['Version'] for i in result):
            result.append(data)
        # Overwrite file info element if already in list but file is signed + valid (preferable)
        elif any(i['Version'] == data['Version'] for i in result) and \
                data['Signature'] == 'OK':
            for index, item in enumerate(result):
                if item['Version'] == data['Version']:
                    result[index] = data

    return result


def get_files_with_infos(dir_path: str) -> List:
    file_infos = []

    for root, _, files in os.walk(dir_path):
        for file in files:
            file_absolute_path = os.path.join(root, file)
            dotnet_file = DotNetPE(file_absolute_path)

            metadata_tables = dotnet_file.existent_metadata_tables()

            entry = {}
            # Get assembly version information
            if 'Assembly' in metadata_tables:
                assembly_version_info = dotnet_file.Assembly.get_assembly_version_information()
                version_info = f'{assembly_version_info.MajorVersion}.{assembly_version_info.MinorVersion}.' \
                               f'{assembly_version_info.BuildNumber}.{assembly_version_info.RevisionNumber}'
                entry['Version'] = version_info

            # Get unmanaged functions
            if 'ImplMap' in metadata_tables:
                unmanaged_functions = '|'.join(dotnet_file.ImplMap.get_unmanaged_functions())
                entry['Unmanaged_functions'] = unmanaged_functions

            # Get compilation timestamp
            comp_time_stamp = datetime.fromtimestamp(dotnet_file.FILE_HEADER.TimeDateStamp).strftime(
                '%Y-%m-%d %H:%M:%S')
            entry['Timestamp'] = comp_time_stamp

            # Get signature information
            pe = lief.parse(file_absolute_path)
            entry['Signature'] = str(pe.verify_signature()).replace('VERIFICATION_FLAGS.', '')

            # Get SHA-256 hash
            with open(file_absolute_path, 'rb') as f:
                entry['Hash'] = sha256(f.read()).hexdigest()

            file_infos.append(entry)

        # Deduplicate and take valid signed file if available. For the rest (invalid/bad/... signature), we
        # don't care
        result = deduplicate(file_infos)

        return result


def print_information_ascending(dir_path: str) -> None:
    file_infos = get_files_with_infos(dir_path)
    file_infos_sorted = sorted(file_infos, key=lambda x: LooseVersion(x['Timestamp']))
```

```
    for file_info in file_infos_sorted:
        output = ', '.join([': '.join(i) for i in file_info.items()])

        print(output)


if __name__ == "__main__":
    print_information_ascending(sys.argv[1])
```

Result:

Version: 2010.1.0.0, Timestamp: 2010-06-25 04:39:04, Signature: NO_SIGNATURE, Hash:
a033d513a3c0e0d0199f06ea4124ab652d5698cecdd276fdb4f4abff6e3602f2
Version: 2011.2.5.2234, Timestamp: 2011-11-04 18:07:31, Signature: NO_SIGNATURE, Hash:
14e799bbdd6517411b41043802a45c85316dc1cc5c61b0f54b043079ec14112b
Version: 2012.1.12.2305, Timestamp: 2012-07-23 13:12:23, Signature: NO_SIGNATURE, Hash:
99cb98b81bca424e109a4c087144fceffdb6e447877fc473ac5be1ea78e55fd9
Version: 2012.1.5.2202, Timestamp: 2012-09-24 16:48:03, Signature: NO_SIGNATURE, Hash:
ad240c7d94308277b241fa060f96f409ecb657a59e639e8f2e79431eb99b9352
Version: 2012.2.13.2381, Timestamp: 2013-01-15 09:40:08, Signature: NO_SIGNATURE, Hash:
acced46ef306939821b0b87671b50a9e93e8eb89c79ba0b81d105ef069d67e21
Version: 2013.2.3.684, Timestamp: 2013-08-20 19:12:32, Signature: OK, Hash:
72f63760a0bd95681d8c78c881e07870b910899c70b52e0e2827317f378773df
Version: 2014.1.1.872, Timestamp: 2013-11-17 03:18:43, Signature: NO_SIGNATURE, Hash:
051e831eaf5bacf840ef6e2af369d6b4b4cd32abd3e10cb1ac80d66e0bb38775
Version: 2013.2.12.2040, Timestamp: 2013-12-04 12:41:57, Signature: NO_SIGNATURE, Hash:
218b25e0507f24e1ab9e231c93d84535d19526241d489985c2770919747f2257
Version: 2014.2.1.1111, Timestamp: 2014-04-08 03:15:00, Signature: OK, Hash:
8d11616c14afc5b2c1b3ee9ee6d46a20cdae31b25e0cd7fbc374b5517b264dd2
Version: 2014.2.11.1229, Timestamp: 2014-08-06 16:48:31, Signature: NO_SIGNATURE, Hash:
dffcc5d7e3ec31a745d59a7f4cd5da1df58ead2d21f57aa04c0b8066f85e4c6a
Version: 2015.1.25300.8197, Unmanaged_functions: CLSIDFromString, Timestamp: 2015-05-27 12:36:44, Signature: NO_SIGNATURE, Hash:
ae95f1b751bf47a83f106e9c019a0051239234f95920581b3f7c8debdc32a5d6
Version: 2015.1.35100.9209, Unmanaged_functions: CLSIDFromString, Timestamp: 2015-12-17 12:51:37, Signature: NO_SIGNATURE, Hash:
d84c2167b03db3760bcf7a76b84534f50910f72a53bc1f5cefcd72784ff75a58
Version: 2016.1.5300.1028, Unmanaged_functions: CLSIDFromString, Timestamp: 2016-05-27 20:04:52, Signature: OK, Hash:
3b4b1d5ebb83fd4d505e0b75673027e80f98861b92b35e0adcd21e2b48ced2d9
Version: 2017.1.100.1628, Unmanaged_functions: CLSIDFromString, Timestamp: 2017-01-26 17:12:52, Signature: NO_SIGNATURE, Hash:
6112ba344a40bc1942ef0b593950542fcc7f67f7ccf1e5274e56e34735efed1b
Version: 2017.1.5300.1698, Unmanaged_functions: CLSIDFromString, Timestamp: 2017-02-21 14:53:54, Signature: NO_SIGNATURE, Hash:
f33c1cb46ad1c82d99f3e92b40e8994aa2d6ec116e9846c131bfcd15de8fa52c
Version: 2017.3.5200.1780, Unmanaged_functions: CLSIDFromString, Timestamp: 2017-08-18 00:46:52, Signature: OK, Hash:
1e229359698c276b2398bfffaa4fb809919e4aaf1c9007d7670866159ecf9ad1
Version: 2018.2.5200.5660, Unmanaged_functions: CLSIDFromString, Timestamp: 2018-08-14 09:52:18, Signature: OK, Hash:
5e4276dc1c9e1e5ef02e75972c59212806dc6a3d13cd2ccafd26af14cb12289b
Version: 2018.2.5200.5675, Unmanaged_functions: CLSIDFromString, Timestamp: 2018-09-25 08:12:02, Signature: NO_SIGNATURE, Hash:
f707ba597c299f503d425867152f6a039b95a401dfa1a6cba8654b65adab78ba
Version: 2018.4.5200.10541, Unmanaged_functions: CLSIDFromString, Timestamp: 2018-11-07 21:05:48, Signature: OK, Hash:
6b295fd9a1c87263613a0d7f23c6c0ece010a038dcce1dd112888bfaff22099e
Version: 2019.2.5200.5206, Unmanaged_functions: CLSIDFromString, Timestamp: 2019-05-18 11:09:22, Signature: OK, Hash:
86901758745e39d3b39bf1373a34fc475e29b3cc418059b54f0ded6bdd5fda01
Version: 2019.4.5200.8890, Unmanaged_functions: CLSIDFromString, Timestamp: 2019-10-10 15:26:39, Signature: OK, Hash:
d3c6785e18fba3749fb785bc313cf8346182f532c59172b69adfb31b96a5d0af
Version: 2019.4.5200.8950, Unmanaged_functions: CLSIDFromString, Timestamp: 2019-11-11 23:34:28, Signature: OK, Hash:
9bee4af53a8cdd7ecabe5d0c77b6011abe887ac516a5a22ad51a058830403690
Version: 2019.4.5200.8996, Unmanaged_functions: CLSIDFromString, Timestamp: 2019-12-09 18:12:09, Signature: OK, Hash:
bb86f66d11592e3312cd03423b754f7337aeebba9204f54b745ed3821de6252d
Version: 2019.4.5200.9001, Unmanaged_functions: CLSIDFromString, Timestamp: 2020-01-03 16:22:18, Signature: OK, Hash:
ae6694fd12679891d95b427444466f186bcdcc79bc0627b590e0cb40de1928ad
Version: 2019.4.5200.9045, Unmanaged_functions: CLSIDFromString, Timestamp: 2020-01-24 12:24:39, Signature: OK, Hash:
9d6285db647e7eeabdb85b409fad61467de1655098fec2e25aeb7770299e9fee
Version: 2019.4.5200.9073, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-03-17 14:08:19, Signature: OK, Hash: db9e63337dacf0c0f1baa06145fd5f1007002c63124f99180f520ac11d551420
Version: 2019.4.5200.9078, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-03-19 10:32:21, Signature: OK, Hash: 0f5d7e6dfdd62c83eb096ba193b5ae394001bac0367454956741 56ead6557589
Version: 2020.2.100.12219, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-03-24 04:00:16, Signature: OK, Hash: dab758bf98d9b36fa057a66cd0284737abf89857b73ca89280267ee7caf62f3b
Version: 2019.4.5200.9083, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-03-24 09:52:34, Signature: OK, Hash: 32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77
Version: 2020.2.100.12299, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-03-25 20:03:48, Signature: OK, Hash: abe22cf0d78836c3ea072daeaf4c5eeaf9c29b6feb597741651979fc8fbd2417
Version: 2020.2.100.12367, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-04-20 10:51:43, Signature: OK, Hash: 2ade1ac8911ad6a23498230a5e119516db47f6e76687f804e2512cc9bcfda2b0
Version: 2020.2.5200.12394, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-04-21 16:53:33, Signature: OK, Hash: 019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134
Version: 2020.2.5300.12432, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-05-11 23:32:40, Signature: OK, Hash: ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6
Version: 2020.4.100.910, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW
 Timestamp: 2020-05-30 11:07:55, Signature: OK, Hash: fcbf4463ffe1f1d3152d65e3b0918d5229997f7b6dcd92ff4e4aec43c8edd890
Version: 2020.4.100.944, Unmanaged_functions:
CLSIDFromString|CloseHandle|AdjustTokenPrivileges|LookupPrivilegeValueW|GetCurrentProcess|OpenProcessToken|InitiateSystemShutdownExW

```
 Timestamp: 2020-06-03 19:02:59, Signature: OK, Hash: 73b11757ef0ec615690156bc09341be30a89d661182f9f4cc0dc2b4e2c4b3a50
Version: 2020.2.15300.12747, Unmanaged_functions: CLSIDFromString, Timestamp: 2020-08-04 15:43:08, Signature: OK, Hash:
018de33dc3a5e8f07207b4349a9d7cf4c59c9d183cc0069014b33e692c54efb5
Version: 2020.2.15300.12766, Unmanaged_functions: CLSIDFromString, Timestamp: 2020-08-11 15:40:55, Signature: OK, Hash:
143632672dcb6ef324343739636b984f5c52ece0e078cfee7c6cac4a3545403a
Version: 2020.2.15300.12901, Unmanaged_functions: CLSIDFromString, Timestamp: 2020-12-11 00:40:07, Signature: OK, Hash:
cc870c07eeb672ab33b6c2be51b173ad5564af5d98bfc02da02367a9e349a76f
Version: 2019.4.5200.9106, Unmanaged_functions: CLSIDFromString, Timestamp: 2020-12-14 03:59:21, Signature: OK, Hash:
8dfe613b00d495fb8905bdf6e1317d3e3ac1f63a626032fa2bdad4750887ee8a
Version: 2020.2.45100.13073, Unmanaged_functions: CLSIDFromString, Timestamp: 2021-01-06 12:53:09, Signature: OK, Hash:
6e91c1cd907c10ba05f6e2c6b37990b1c061a06803d50d34521efa0f4796f7dd
Version: 2020.2.45100.13097, Unmanaged_functions: CLSIDFromString, Timestamp: 2021-01-14 13:12:20, Signature: OK, Hash:
55e4ec31ae9bd625f28d924ee3cca3c97bc816d249709ba5a29607608bebd7e9
Version: 2020.2.55200.29741, Unmanaged_functions: CLSIDFromString, Timestamp: 2021-03-01 20:24:04, Signature: OK, Hash:
74ab5f1405f89462b1e8de99f87ece9e474ab1a2b5d2f59f19bfa822d59405ed
Version: 2020.2.65100.50257, Unmanaged_functions: CLSIDFromString, Timestamp: 2021-08-13 16:14:12, Signature: OK, Hash:
d0c79a03a58981e1850e4ec5bc860c65f6042df7b1a4539470f5da86b4c6ac01
Version: 2016.2.5300.959, Unmanaged_functions: CLSIDFromString, Timestamp: 2022-05-09 22:08:23, Signature: NO_SIGNATURE, Hash:
14426746efc5a8e3ee45749df2055efa70f680de8832e0bb1ea734efeefde935
```

Timeline evaluation

For a better overview, let's put the data in a table:

Orion DLL version	Backdoored	Compilation timestamp	Signed with valid signature	SHA-256
2010.1.0.0	No	2010-06-25 04:39:04	No	a033d513a3c0e0d0199f06ea4124ab652d5698cecdd276fdb4f4abff6e3602f2
2011.2.5.2234	No	2011-11-04 18:07:31	No	14e799bbdd6517411b41043802a45c85316dc1cc5c61b0f54b043079ec14112b
2012.1.12.2305	No	2012-07-23 13:12:23	No	99cb98b81bca424e109a4c087144fceffdb6e447877fc473ac5be1ea78e55fd9
2012.1.5.2202	No	2012-09-24 16:48:03	No	ad240c7d94308277b241fa060f96f409ecb657a59e639e8f2e79431eb99b9352
2012.2.13.2381	No	2013-01-15 09:40:08	No	acced46ef306939821b0b87671b50a9e93e8eb89c79ba0b81d105ef069d67e21
2013.2.3.684	No	2013-08-20 19:12:32	Yes	72f63760a0bd95681d8c78c881e07870b910899c70b52e0e2827317f378773df
2014.1.1.872	No	2013-11-17 03:18:43	No	051e831eaf5bacf840ef6e2af369d6b4b4cd32abd3e10cb1ac80d66e0bb38775
2013.2.12.2040	No	2013-12-04 12:41:57	No	218b25e0507f24e1ab9e231c93d84535d19526241d489985c2770919747f2257
2014.2.1.1111	No	2014-04-08 03:15:00	Yes	8d11616c14afc5b2c1b3ee9ee6d46a20cdae31b25e0cd7fbc374b5517b264dd2
2014.2.11.1229	No	2014-08-06 16:48:31	No	dffcc5d7e3ec31a745d59a7f4cd5da1df58ead2d21f57aa04c0b8066f85e4c6a
2015.1.25300.8197	No	2015-05-27 12:36:44	No	ae95f1b751bf47a83f106e9c019a0051239234f95920581b3f7c8debdc32a5d6
2015.1.35100.9209	No	2015-12-17 12:51:37	No	d84c2167b03db3760bcf7a76b84534f50910f72a53bc1f5cefcd72784ff75a58
2016.1.5300.1028	No	2016-05-27 20:04:52	Yes	3b4b1d5ebb83fd4d505e0b75673027e80f98861b92b35e0adcd21e2b48ced2d9
2017.1.100.1628	No	2017-01-26 17:12:52	No	6112ba344a40bc1942ef0b593950542fcc7f67f7ccf1e5274e56e34735efed1b
2017.1.5300.1698	No	2017-02-21 14:53:54	No	f33c1cb46ad1c82d99f3e92b40e8994aa2d6ec116e9846c131bfcd15de8fa52c
2017.3.5200.1780	No	2017-08-18 00:46:52	Yes	1e229359698c276b2398bfffaa4fb809919e4aaf1c9007d7670866159ecf9ad1
2018.2.5200.5660	No	2018-08-14 09:52:18	Yes	5e4276dc1c9e1e5ef02e75972c59212806dc6a3d13cd2ccafd26af14cb12289b

Orion DLL version	Backdoored	Compilation timestamp	Signed with valid signature	SHA-256
2018.2.5200.5675	No	2018-09-25 08:12:02	No	f707ba597c299f503d425867152f6a039b95a401dfa1a6cba8654b65adab78ba
2018.4.5200.10541	No	2018-11-07 21:05:48	Yes	6b295fd9a1c87263613a0d7f23c6c0ece010a038dcce1dd112888bfaff22099e
2019.2.5200.5206	No	2019-05-18 11:09:22	Yes	86901758745e39d3b39bf1373a34fc475e29b3cc418059b54f0ded6bdd5fda01
2019.4.5200.8890	No	**2019-10-10 15:26:39**	Yes	d3c6785e18fba3749fb785bc313cf8346182f532c59172b69adfb31b96a5d0af
2019.4.5200.8950	No	2019-11-11 23:34:28	Yes	9bee4af53a8cdd7ecabe5d0c77b6011abe887ac516a5a22ad51a058830403690
2019.4.5200.8996	No	2019-12-09 18:12:09	Yes	bb86f66d11592e3312cd03423b754f7337aeebba9204f54b745ed3821de6252d
2019.4.5200.9001	No	2020-01-03 16:22:18	Yes	ae6694fd12679891d95b427444466f186bcdcc79bc0627b590e0cb40de1928ad
2019.4.5200.9045	No	2020-01-24 12:24:39	Yes	9d6285db647e7eeabdb85b409fad61467de1655098fec2e25aeb7770299e9fee
2019.4.5200.9073	Yes	**2020-03-17 14:08:19**	Yes	db9e63337dacf0c0f1baa06145fd5f1007002c63124f99180f520ac11d551420
2019.4.5200.9078	Yes	**2020-03-19 10:32:21**	Yes	0f5d7e6dfdd62c83eb096ba193b5ae394001bac036745495674156ead6557589
2020.2.100.12219	Yes	**2020-03-24 04:00:16**	Yes	dab758bf98d9b36fa057a66cd0284737abf89857b73ca89280267ee7caf62f3b
2019.4.5200.9083	Yes	**2020-03-24 09:52:34**	Yes	32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c7
2020.2.100.12299	Yes	**2020-03-25 20:03:48**	Yes	abe22cf0d78836c3ea072daeaf4c5eeaf9c29b6feb597741651979fc8fbd2417
2020.2.100.12367	Yes	**2020-04-20 10:51:43**	Yes	2ade1ac8911ad6a23498230a5e119516db47f6e76687f804e2512cc9bcfda2b0
2020.2.5200.12394	Yes	**2020-04-21 16:53:33**	Yes	019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134
2020.2.5300.12432	Yes	**2020-05-11 23:32:40**	Yes	ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6
2020.4.100.910	Yes	**2020-05-30 11:07:55**	Yes	fcbf4463ffe1f1d3152d65e3b0918d5229997f7b6dcd92ff4e4aec43c8edd890
2020.4.100.944	Yes	**2020-06-03 19:02:59**	Yes	73b11757ef0ec615690156bc09341be30a89d661182f9f4cc0dc2b4e2c4b3a50
2020.2.15300.12747	No	2020-08-04 15:43:08	Yes	018de33dc3a5e8f07207b4349a9d7cf4c59c9d183cc0069014b33e692c54efb5
2020.2.15300.12766	No	2020-08-11 15:40:55	Yes	143632672dcb6ef324343739636b984f5c52ece0e078cfee7c6cac4a3545403a
2020.2.15300.12901	No	2020-12-11 00:40:07	Yes	cc870c07eeb672ab33b6c2be51b173ad5564af5d98bfc02da02367a9e349a76f
2019.4.5200.9106	No	2020-12-14 03:59:21	Yes	8dfe613b00d495fb8905bdf6e1317d3e3ac1f63a626032fa2bdad4750887ee8a
2020.2.45100.13073	No	2021-01-06 12:53:09	Yes	6e91c1cd907c10ba05f6e2c6b37990b1c061a06803d50d34521efa0f4796f7dd
2020.2.45100.13097	No	2021-01-14 13:12:20	Yes	55e4ec31ae9bd625f28d924ee3cca3c97bc816d249709ba5a29607608bebd7e9
2020.2.55200.29741	No	2021-03-01 20:24:04	Yes	74ab5f1405f89462b1e8de99f87ece9e474ab1a2b5d2f59f19bfa822d59405ed

Orion DLL version	Backdoored	Compilation timestamp	Signed with valid signature	SHA-256
2020.2.65100.50257	No	2021-08-13 16:14:12	Yes	d0c79a03a58981e1850e4ec5bc860c65f6042df7b1a4539470f5da86b4c6ac01
2016.2.5300.959	No	2022-05-09 22:08:23	No	14426746efc5a8e3ee45749df2055efa70f680de8832e0bb1ea734efeefde935

The entries marked in red are the Sunburst samples, the one marked in blue is the Sunburst sample which contains only test code. The last entry can be ignored since the sample dataset contains only `2016.2` files with invalid signatures. These files cannot be considered trustworthy as shown by the compilation timestamp of the sample. However, all files that are important to us have valid signatures.

From the data, we can make some observations, which I will describe in more detail later:

1. The attackers created `10 backdoored versions` over the course of `~3 months`
2. The `first` backdoored version was spread on `2020-03-17` , the `last` on `2020-06-03`
3. The `first` backdoored `Orion version` was `2019.4.5200.9073` , the `last` was `2020.4.100.944`

Point 1

The first five backdoored versions are created at a higher frequency than any legitimate updates. There are only days in between them or have even been created on the same day. This raises suspicions that the attackers could have been able to trigger the creation of Orion updates. They may not just have waited until the SolarWinds developers released updates to inject Sunburst, as one might conclude from reading the Sunspot analysis. Having full control over the entire development pipeline makes it much easier to coordinate such a crucial mission. While this is a noisier approach than passively waiting for official Orion releases, it also increases the chances of success. However, this is only a theory and may not be true as I do not have a complete sample set as indicated in the next chapter.

Point 2

Our data shows that the first backdoored sample was created on `March 17, 2020` , whereas the SolarWinds investigation shows `February 20, 2020` . This means our sample data set is unfortunately incomplete, so the assumption in point 1 may also be incorrect. The last backdoored release on `June 3, 2020` coincides with the statement from SolarWinds.

Point 3

This is the most interesting point cause of the two backdoored Orion versions `2020.4.100.910` and `2020.4.100.944` that stand out. These versions are not shown to be affected as a result of the SolarWinds investigation. This specific version `2020.4` is not even shown on the Orion release notes page. We can also see that after the attackers have stopped to spread Sunburst samples, the `2020` version continues with `2020.2` by the SolarWinds developers. This could mean that the attackers somehow created the `2020.4` branch.

It goes without saying that all the hypotheses made in points 1 and 3 must be taken with a grain of salt. Since I do not have a complete set of Orion samples and no insights into any non-public investigation results, this is only speculation.

Conclusion

With the help of `dotnetfile` you can pull information from .NET header's **on scale**. As I showed on the Sunburst example, you can create a version timeline of several samples in no time compared to manually collecting the data. One of the sources of information can be the `ImplMap` table where unmanaged functions are located. In this article, I just gave a very simple example of intelligence gathering. You can do much more with this library in terms of malware detection and threat hunting. It just takes a little creativity and imagination, there is a lot of information to discover in the CLR header.

Sample set hashes (SHA-256)

018de33dc3a5e8f07207b4349a9d7cf4c59c9d183cc0069014b33e692c54efb5
019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134
019af890df78f69847b4493a536559e69f1734e40bf08bb46ea98d9682389829
02dfee607de7969d23551b8f9f5d605ec95cfdf5b7822376f8b4df456a6ca8c7
051e831eaf5bacf840ef6e2af369d6b4b4cd32abd3e10cb1ac80d66e0bb38775
0dbbf3949dfbeb53767aab3f8fe0534e686b955b43367700808c8c44e22d8e00
0f5d7e6dfdd62c83eb096ba193b5ae394001bac036745495674156ead6557589
13bfd97ccb585c082495242ebd73fe779576920b0f0ab3c5adea13f7d75f6aac
143632672dcb6ef3243437396 36b984f5c52ece0e078cfee7c6cac4a3545403a
14426746efc5a8e3ee45749df2055efa70f680de8832e0bb1ea734efeefde935
14e799bbdd6517411b41043802a45c85316dc1cc5c61b0f54b043079ec14112b
1bb7c3edb5a8e3c6d5f77e57a84b80a11b08ed10c0ee2a5ef14856e38872b917
1e229359698c276b2398bfffaa4fb809919e4aaf1c9007d7670866159ecf9ad1
218b25e0507f24e1ab9e231c93d84535d19526241d489985c2770919747f2257
23428bd54c823125553463dc4d1faf6e2c03f09d6652d61ae4e49f93d0de3ec5
2435fd31855898ed41d194570febc50f3027aa584298ff264cd99f8c84b0ec57
2881406f09bb21bf62abfe2541eb2bebb1db5209e8faf4490eb23b398ad80438
2ade1ac8911ad6a23498230a5e119516db47f6e76687f804e2512cc9bcfda2b0
317a0c715cbd83061e10cb7a1eaccde4786ab848a8881e91f960d7606b9352c6
32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77
3b4b1d5ebb83fd4d505e0b75673027e80f98861b92b35e0adcd21e2b48ced2d9
3e96a5d3307158a0c1af97e154f797286ffe77590acf6e051c9aa6b6508c1b9a
42e73c85b07d89956e94db832d6501c823ae00684c50d9c9163194357c3dd3ed
468d425643350410c1cb43841244f8c0accad58d6e6bc99239dac7aac4e6fd90
46a64842fc7e1137f2e344df46523710b5420d9625606454004f24c3faeb1051
502857c523178fb69190984271f5c6948ff17b0db85aa3c23b2a68a09c0bddc8
51ae0f0648cb8db221fb46b95bf709e5fb10dc65be8156f48991aa59a4405db6
53bfbe32676f6990a49bb2a001eaa78a8cc2ba9c0f3eb484337c1d7a2f56ce72
55e4ec31ae9bd625f28d924ee3cca3c97bc816d249709ba5a29607608bebd7e9
5b448751ea1c845ce3f9b979799369ba44585e72199c1252b16e7e5b8dd588fd
5b931138fb32e03f651c6c8c6c4fc40c9b78441e04f7d6026394d210749ac3d9
5d6cd94a87f0154fa2deaf2ad0931394256f26ca6278a3549596f89b053b0ef4
5e4276dc1c9e1e5ef02e75972c59212806dc6a3d13cd2ccafd26af14cb12289b
608d1994238fea66c22e45817fb2c24f7fdcf8d86adfbd015f8dab05c3f699d9
6112ba344a40bc1942ef0b593950542fcc7f67f7ccf1e5274e56e34735efed1b
6b295fd9a1c87263613a0d7f23c6c0ece010a038dcce1dd112888bfaff22099e
6bc622ceb9729abd6b27d228efe9b9741a752aa6cff5f81216297575029e4e08
6e91c1cd907c10ba05f6e2c6b37990b1c061a06803d50d34521efa0f4796f7dd
6f9bfc4379a904e34aa0d8c24fd4256cdd0b4a51e6b1f7bf0fa75a7bc2c79a0a
72f63760a0bd95681d8c78c881e07870b910899c70b52e0e2827317f378773df
73b11757ef0ec615690156bc09341be30a89d661182f9f4cc0dc2b4e2c4b3a50
74ab5f1405f89462b1e8de99f87ece9e474ab1a2b5d2f59f19bfa822d59405ed
84dc86c1707edf58a795063d7ca32a063b42b3e6bc397cfce05dc56a4b7f7db4
86901758745e39d3b39bf1373a34fc475e29b3cc418059b54f0ded6bdd5fda01
8d11616c14afc5b2c1b3ee9ee6d46a20cdae31b25e0cd7fbc374b5517b264dd2
8dfe613b00d495fb8905bdf6e1317d3e3ac1f63a626032fa2bdad4750887ee8a
9590b202876f06678d4af637c303aaecbd45ebde8abd3ec0eaa5b7d88c54bce6
99cb98b81bca424e109a4c087144fceffdb6e447877fc473ac5be1ea78e55fd9
9bee4af53a8cdd7ecabe5d0c77b6011abe887ac516a5a22ad51a058830403690
9d6285db647e7eeabdb85b409fad61467de1655098fec2e25aeb7770299e9fee
a033d513a3c0e0d0199f06ea4124ab652d5698cecdd276fdb4f4abff6e3602f2
a25cadd48d70f6ea0c4a241d99c5241269e6faccb4054e62d16784640f8e53bc
a5813de9e453c86c9ea07668deb090691d568cb0dad80197319a2d5657ee170c
abe22cf0d78836c3ea072daeaf4c5eeaf9c29b6feb597741651979fc8fbd2417
acced46ef306939821b0b87671b50a9e93e8eb89c79ba0b81d105ef069d67e21
ad240c7d94308277b241fa060f96f409ecb657a59e639e8f2e79431eb99b9352
ae6694fd12679891d95b427444466f186bcdcc79bc0627b590e0cb40de1928ad
ae95f1b751bf47a83f106e9c019a0051239234f95920581b3f7c8debdc32a5d6
afb8f6d5c7e3b55a79257d0c4736fec6e52abfa3e1540bcfd16b954bd422b33c
b3b7107f7784f728c3339e679ce153897dbd44bde4799eb10f32972813169f2a
b9ce678f9daf32c526211edea88b5ec104538c75fad13767ea44309e9f81dbfc
b9defa16d1aa92d85d1d5d47339c999eee42aa3b9ada5dd4d5a158efcadd509a
bb86f66d11592e3312cd03423b754f7337aeebba9204f54b745ed3821de6252d
c419b50e1fd87884019c83b7fc4f8096adc9f2c7da15409d0f77613e28d81103
c5e76c932daf2c508547d5a910172d919dbe69d62808d0d94ee32af860708ba8
cbc24292a8b09dde6ff95d4a5c00c5e741d85cfc26510fabafbc14565c623966
cc870c07eeb672ab33b6c2be51b173ad5564af5d98bfc02da02367a9e349a76f
ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6
ced0a4be5059c986965b070edd8a1706657f834629a31b599baef5ddca6e375d
d0c79a03a58981e1850e4ec5bc860c65f6042df7b1a4539470f5da86b4c6ac01
d3c6785e18fba3749fb785bc313cf8346182f532c59172b69adfb31b96a5d0af
d84c2167b03db3760bcf7a76b84534f50910f72a53bc1f5cefcd72784ff75a58
dab758bf98d9b36fa057a66cd0284737abf89857b73ca89280267ee7caf62f3b
db9e63337dacf0c0f1baa06145fd5f1007002c63124f99180f520ac11d551420
dcbf229f582756bf3f730efa3f460c4381b8e042bbfd4aa1ebe5c763f7c27546
dee8216b2e7b04754501ebc3c19e5351c60ba83f5664bf6172589a94e4ed3592
dffcc5d7e3ec31a745d59a7f4cd5da1df58ead2d21f57aa04c0b8066f85e4c6a
e60f0fd87784f7d3c4b17331676e570554616cdae40a1e4503932f5680820cf7
f33c1cb46ad1c82d99f3e92b40e8994aa2d6ec116e9846c131bfcd15de8fa52c

```
f3a622b84e632e255797fa2f7da9de8e05bc523931da6e9327dc1db8171d69aa
f707ba597c299f503d425867152f6a039b95a401dfa1a6cba8654b65adab78ba
fcbf4463ffe1f1d3152d65e3b0918d5229997f7b6dcd92ff4e4aec43c8edd890
```