

CrystalBit / Apple Double DLL Hijack

 blog.morphisec.com/crystalbit-apple-double-dll-hijack



- [Tweet](#)

•



As part of a rapid change in the work environment during the COVID-19 pandemic, Morphisec Labs has been tracking the change in the attack trend landscape. This has included the evolution of adware, PUA, and fraudulent software bundle delivery beyond a consumer problem into a significant attack vector on enterprise employees.

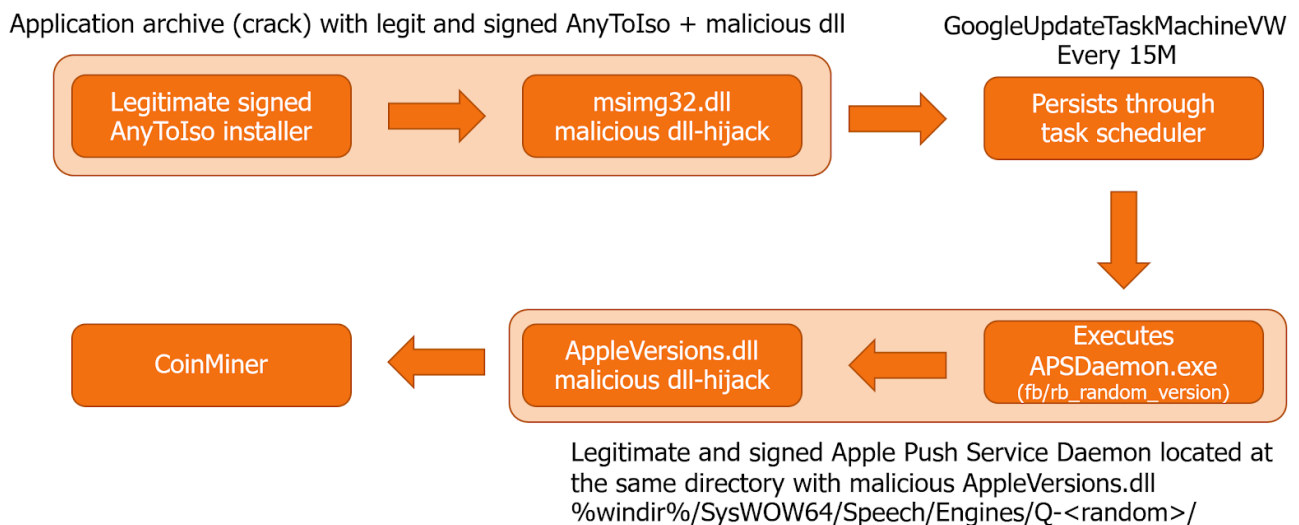
In this vein, Morphisec Labs has seen a more than 800 percent increase in preventing attacks from adware and fraudulent software bundles among protected enterprises.

In this blog, we will technically dive into one significant attack campaign example that happened in the second week of May. During this attack, the adversary abuses two legitimate vendor applications, such as CrystalBit and Apple, as part of a **dll double hijack** attack chain that starts with a fraudulent software bundle and eventually leads to a persistent miner and in some cases spyware deployment.

The abuse of the Apple push notification executable (APSDaemon.exe) is certainly not new. Over the course of more than a year, adversaries have deployed a legitimate and signed copy of the application together with a malicious AppleVersions.dll that is soon loaded by the daemon. In most cases, the deployment was part of a second stage of an attack and rarely have been seen as part of an infiltration stage.

What caught our attention is the use of similar techniques by abusing AnyToIso and CrystalBit software as part of the first delivery stage of the attack.

DLL HIJACK Technical Overview







The attack chain consists of:

1. Downloading a software bundle from a fraudulent site (DVD plugins, browsers, Excel plugins, codecs, etc...).





Format: [random letters and digits]{7,8}_SETUP.zip

- Elevated Execution of a signed AnyTolso / CrystalBit application by the victim, which leads to a malicious msimg32.dll hijack (the malicious DLL is bundled together with the executable)

Name	Date modified
 DevExpress.dll	21/05/2020 04:08
 msimg32.dll	21/05/2020 04:08
 QIPcap64.dll	21/05/2020 04:08
 RBGO09XG_SETUP.exe	17/11/2019 08:57

- Msimg32.dll writes a number of files that include the Apple push application and the malicious AppleVersions.dll to a “super” hidden directory under %windir%/SysWOW64/Speech/Engines/. In some cases it also writes data.dll. Both the directory name within the Speech/Engines and the APSDaemon.exe file name are randomized under given constraints
 - The APSDaemon.exe new name starts with “fb_” or “rb_”, (“fb_<number>.exe”).
 - The directory name starts with “Q-”.

« SysWOW64 > Speech > Engines > Q-1-19-84

Name
 AppleVersions.dll
 FD_1.4.51.27.exe
 msvcp100.dll
 msvcr100.dll

-

Msimg32.dll creates a scheduled task for persistence, in most cases we have observed a GoogleUpdateTask that will execute every 15 minutes. Note: the scheduled task is created with system privileges, you will need system privs to view it.

Note: to access the folder you will need to disable hidden privileges (attrib -h -s).

```

1  <?xml version="1.0" encoding="UTF-16"?>
2  <Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
3    <RegistrationInfo>
4      <Author>SYSTEM</Author>
5      <Version>1.4.95.73</Version>
6      <Description>Make sure that your Google software is always up-to-date. If you disab
7      <URI>\Microsoft\Windows\Google\GoogleUpdateTaskMachineVW</URI>
8      <SecurityDescriptor>O:SYD:P(A;;;GA;;;S-1-5-18)</SecurityDescriptor>
9    </RegistrationInfo>
10   <Triggers>
11     <TimeTrigger>
12       <Repetition>
13         <Interval>PT15M</Interval>
14         <StopAtDurationEnd>>false</StopAtDurationEnd>
15       </Repetition>
16       <StartBoundary>1999-01-01T20:36:23</StartBoundary>
17       <Enabled>>true</Enabled>
18       <RandomDelay>PT13S</RandomDelay>
19     </TimeTrigger>
20   </Triggers>
21   <Principals>
22     <Principal id="Author">
23       <UserId>S-1-5-18</UserId>
24       <RunLevel>HighestAvailable</RunLevel>
25     </Principal>
26   </Principals>
27   <Settings>
28   <Actions Context="Author">
29     <Exec>
30       <Command>C:\WINDOWS\SysWOW64\Speech\Engines\Q-1-97-11\FD_1.4.19.89.exe</Command>
31     </Exec>

```

CoinLoader Shellcode

VirusTotal and other 3rd party tools have classified some of the AppleVersions.dll versions in a very generic way, while some dll versions have got a clear CoinLoader classification, yet with a relatively low detection ratio. We have decided to investigate some of those and identified interesting techniques to extract modules and functions from within the memory (known but rare).

More specifically, their shellcode implemented a variation of the **Fowler–Noll–Vo** hash algorithm to compare module names and function names during the iteration over the Process environment block structure. This significantly increased their chances to evade security vendors that are looking for the regular hash signatures in memory (all leading vendors are looking for the ROR-13 hash patterns as those are implemented as part of the default code injection framework unless the framework is recompiled).


```

226 while ( 1 )
227 {
228     v9 = *((_DWORD *)func_name_rva - 1);
229     func_name_rva -= 4;
230     func_name = (char *)baseAddress + v9;
231     --name_idx;
232     v11 = 0x811C9DC5; // seed
233     chr = *func_name;
234     v13 = (int)(func_name + 1);
235     if ( chr )
236     {
237         do
238         {
239             ++v13;
240             v11 = 0x1000193 * (v11 ^ chr); // fnv_32_prime
241             chr = *(_BYTE *)(v13 - 1);
242         }
243         while ( chr );
244         if ( v11 == 0x992ED028 ) // fnv hash of lstrlenA
245             break;
246     }
247     if ( !name_idx )
248     {
249         v4 = v16;
250         goto LABEL_13;
251     }
252 }
253 lstrlenA = (int (__stdcall *)(int *))((char *)baseAddress
254                                     + *((_DWORD *)((char *)baseAddress
255                                                     + 4
256                                                     * *(unsigned __int16 *)((char *)baseAddress
257                                                         + 2 * name_idx
258                                                         + v17->AddressOfNameOrdinals)
259                                                         + v17->AddressOfFunctions)));
260 LABEL_16:
261 dword_101C8740 = (int)lstrlenA;

```

Conclusions

Malware has become much more evasive without relation to its **type or category**. This evasiveness manifests itself through whitelisting bypass, fileless techniques and in-memory execution. Looking for suspicious memory patterns is not enough and will not hold for long, which makes prevention a clear necessity.

COVID-19 is the perfect time for adversaries to go hunt for new and less protected environments, WFH (work from home) is such an environment without a proper network protection stack, without proper hardening and without proper IT management and enforcement.

Morphisec protects such environments without applying any sort of detection by executing Moving Target Defense against the same adversaries.

Artifacts

AppleVersions.dll

be2e196f2920766f4bd63c1c8b566f3546e3fa94e49a1d9ccbc7d5eab54bca2c

171efee993d8f3b6511cf2db29c5f73179706835a23bdd56e19760a5178763d7

5e0299afdc9a0a37b364be15520b274565a6eb3894aefb6d9a4fd922a045e919
99b8b6c1063dcdfd1e48819047543d03a2c3b1668f34ded03caae011ebeaf077
d5511bbf5c3242e9bf286cbd158f9b29f7f279539a2f4b6a0ffb5551bcf6ddab
90fce0aec20449e9dc200bd661fa27cd76ffb64a1d998753e9022d618cd6f3dc
288a5cbc213c992991a2c26827bf6dd0c49570797a8be681db3bffc8169302b1
84bfa4c1d0a5fdb3bb92e5df208aa4380147edb20ae5138482d46a4f286ed16c
9e1d1ad0ed4d3a6c91d760140c63f1fc667ef26c5235fe3cac9ceaa412b3dfe7
0607b96955d0d892f7ed07c238a608d88b4af8e0da3e23a74257788c1642f3d5
422d0124825e9aa80d947440de9924a9f30fa06d110f9b6f77a183130892584e
048f0a2ab98627a8c9181e5d867b8114e648afa718ed11830c29882e07556ec4
05f82d05e2281a431ab92056718b11073fb0a2ea65dc2ab646f38b5b4ef88285
36715e5bd086735793d2f2f708cce7e8bd2e9962d4a64e38f79b5b78a87f1a5e
5e78e29357a07517ff907d9cd10b9ec41f5132d84e55aa3ba3e9b95ee854f507
b336e88760604e2bc6e11b64357f0ca9ce940860e25a2e0eea4b16f2bb01b11d
b0fb503452c986755b12466bb7e078e9c1a7dd94b3bc918aabd905dcc831e4ed
920bb63ff5674de12508caa1b6e54e8dfe7c4c4c92e790fd69d3e79ed069cd35
f04611b23d9eade1378205f68e16d68196121ff183420a191b71553a0b210d4d
9bd4695e1c47a07c7d9667093580f99ad8d5a83090d537dcabb235b06bb88d8e
6bc08768a5c3136de42e05d8051c2d17a7c65712363f75d8e56c015ac023a3f0
fb3039cbc5dc39017d67de2b39971d30c836efd2370b3a16177979dbbexbbb88a
a14837ba556522a151b507360baaf42285cadaf75bdb0b15f86ada031833e27a
088388c4aed1eecd75d93d95eb41cfea563cbcf2ef324e2d1817797da4dc0215
26a112534dd567f95d2f78f15c6a0d41507c32b190a6023eeaea11e20bc33f3a
9de38aa482828653edf5d1cc4a6631af79c95082ba9cdf67b7929d33bc7e42c1
843207d8bf91d6345943941292ade06a3f3d78d3c7a381d03a06618dc9690056

2738dfb0ce4f24cdd866da1ae699667088c4c1b89259408ebc1eed3c330faa0
6f887d71f64c3cea6ae2962abf4c5c6884574d20c2475957d0c91f299e82afd8
0888b450848000dbf4cdd35de0755b8ca5388718757d7c3de0755f5d00c9fbf8
ada8b31410bfa6feb10736607ad289f8fa938cd82387b32decb8d34d4be82814

Msimg32.dll

6dc8f7a18e1fd63e741635670ce8b1a96149f50cb10b672edf2f58f3cbb9898c
6e1dc8519b4663e14116363029be7d290d22cea4198884c79a9b3c036ff8c3d5
4109fc98007e41d7cf63547c29b6c558c67dec938d280c8ed091eafdccce2732
f0d4f8c304032a1087d8a70978943bf7fcd9d69df514e2efa03ddc264403b9ff
f0b9910df11b920d27da387eb4cabe5db21e07166aaf530fd877698225b7588e
a86158e10fda310befe2678e77fb82742e7e486ae153f9792dc630625efcf1e9
8da42709a5e6ab3dd66f31eb76efed0026d0d580849e44da1bbc3315170566fa
bd9713d2710851854cc3ec33807639df450d0a7665d238ebff14a8b434426379
6dc8f7a18e1fd63e741635670ce8b1a96149f50cb10b672edf2f58f3cbb9898c
aeb99b84f571b460d6e05b2b7b644cb6c11e891b625390963e57ec5470ea15b

NEW THREAT INDEX
**MORPHISEC'S 2020 WFH EMPLOYEE
CYBERSECURITY THREAT INDEX**
Learn what challenges are unique to work-from-home employee security.
DOWNLOAD NOW

[Contact SalesInquire via Azure](#)