Hunting for Suspicious Usage of Background Intelligent Transfer Service (BITS)

blog.menasec.net/2021/05/hunting-for-suspicious-usage-of.html

BITS Overview

Background Intelligent Transfer Service (BITS) is used by programmers and system administrators to download files from or upload files to HTTP web servers and SMB file shares.

BITS will take the cost of the transfer into consideration, as well as the network usage so that the user's foreground work has as little impact as possible. BITS also handles network interruptions, pausing and automatically resuming transfers, even after a reboot, which makes it a very good candidate for implant Command and Control standard tasks (download, upload and ex-filtration).

BITS includes PowerShell cmdlets for creating and managing transfers as well as the BitsAdmin command-line utility.

BITS is composed of a Client (i.e. bitsadmin, powershell) loading Bitsproxy.dll, qmgrprxy.dll or Microsoft.BackgroundIntelligentTransfer.Management.Interop.dll and a Server (svchost.exe with the process's command-line value contains the keyword "BITS" and hosting the service DLL qmgr.dll):

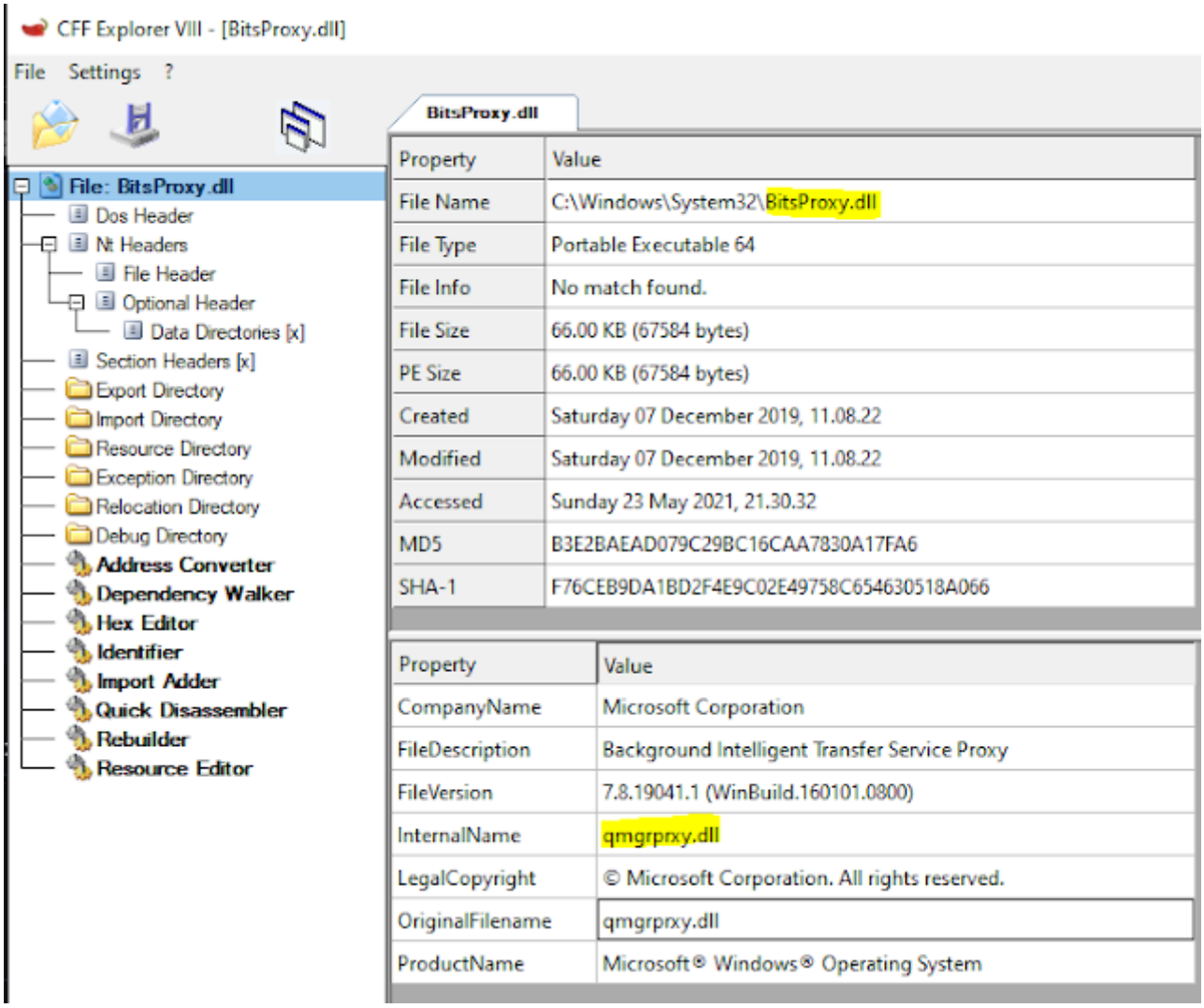


Figure 1 - BITS Client

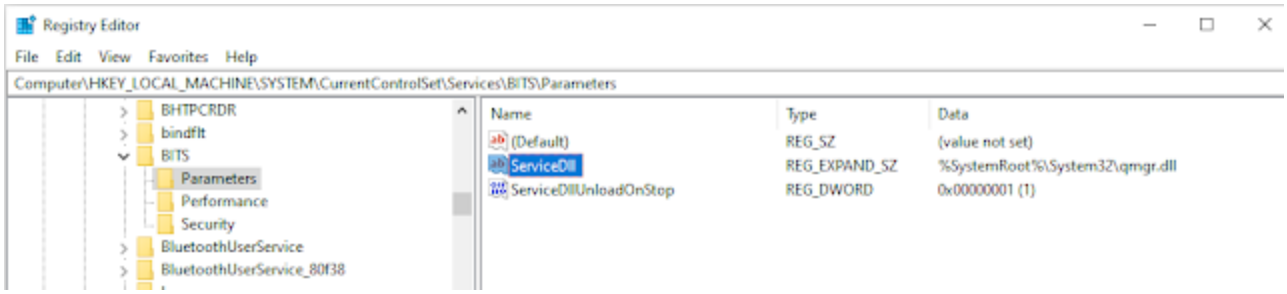


Figure 2 - BITS Server

Communication between the BITS client and server is performed via RPC, and the IBackgroundCopyManager is the main BITS interface used to enumerate or create new BITS Jobs:

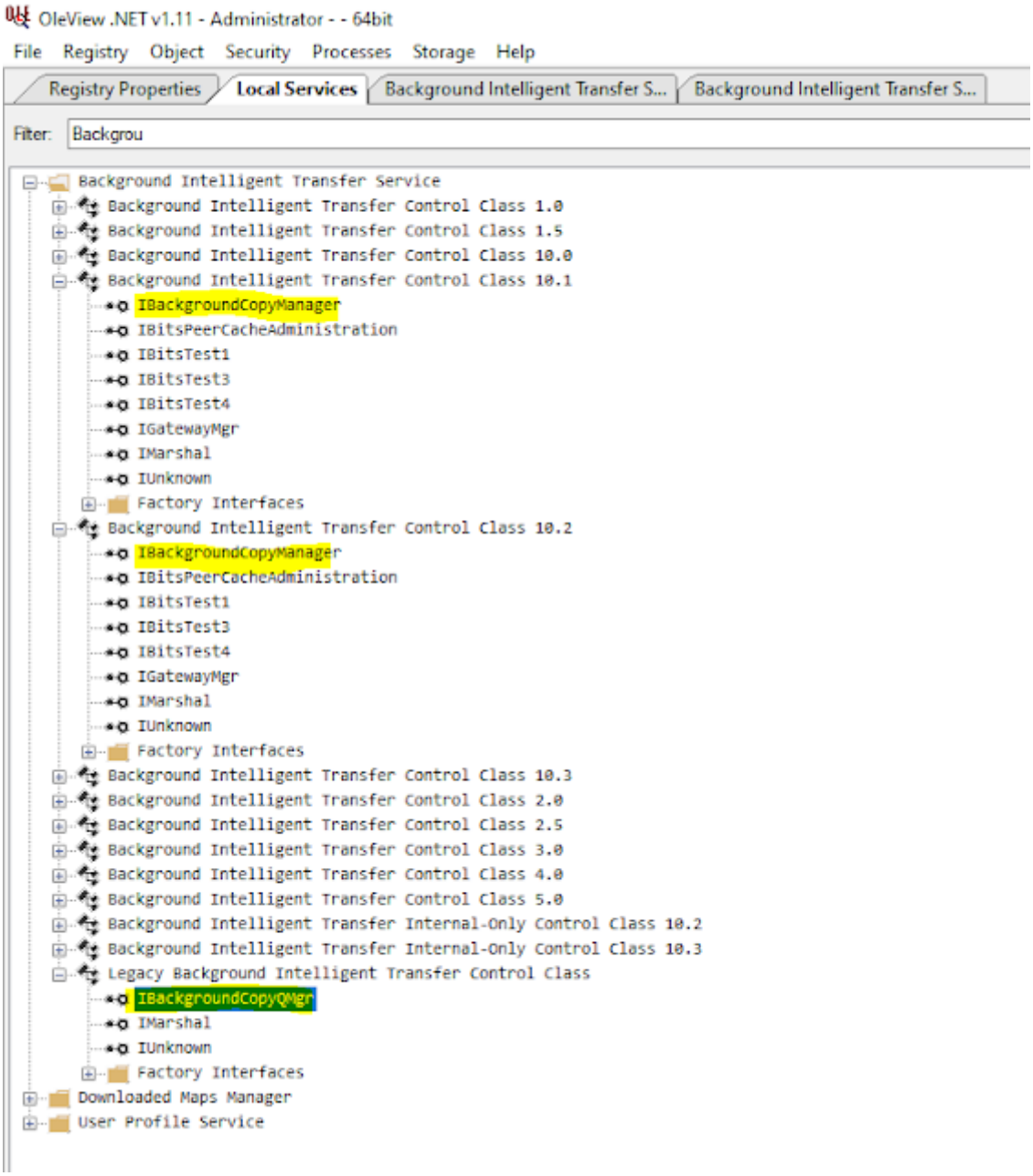
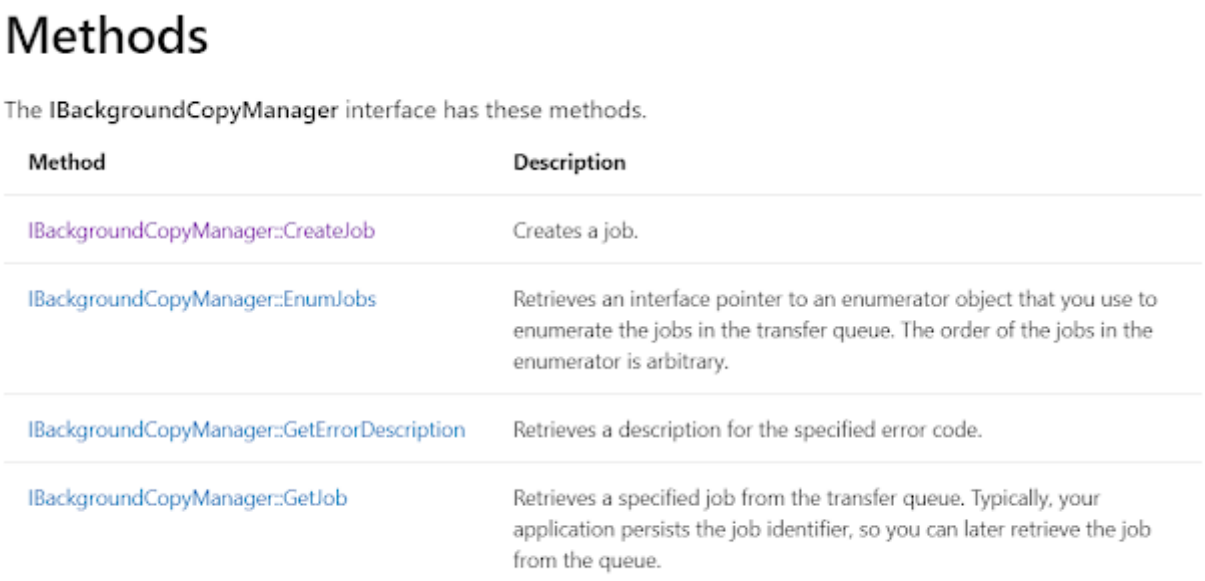


Figure 3 - OleView of the BITS service exposed interfaces

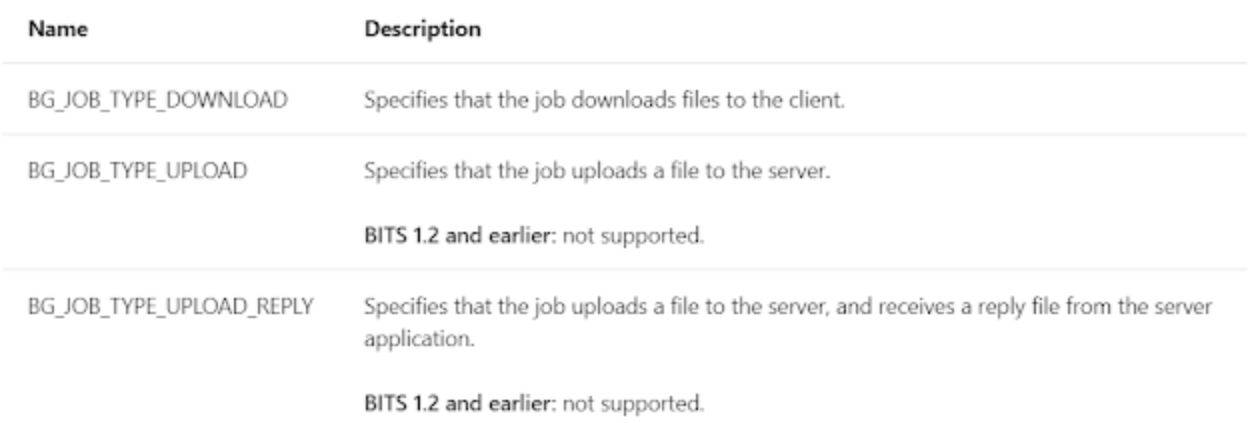


Methods

The **IBackgroundCopyManager** interface has these methods.

Method	Description
IBackgroundCopyManager::CreateJob	Creates a job.
IBackgroundCopyManager::EnumJobs	Retrieves an interface pointer to an enumerator object that you use to enumerate the jobs in the transfer queue. The order of the jobs in the enumerator is arbitrary.
IBackgroundCopyManager::GetErrorDescription	Retrieves a description for the specified error code.
IBackgroundCopyManager::GetJob	Retrieves a specified job from the transfer queue. Typically, your application persists the job identifier, so you can later retrieve the job from the queue.

Figure 4 - BITS IBackgroundCopyManager Interface exposed Methods

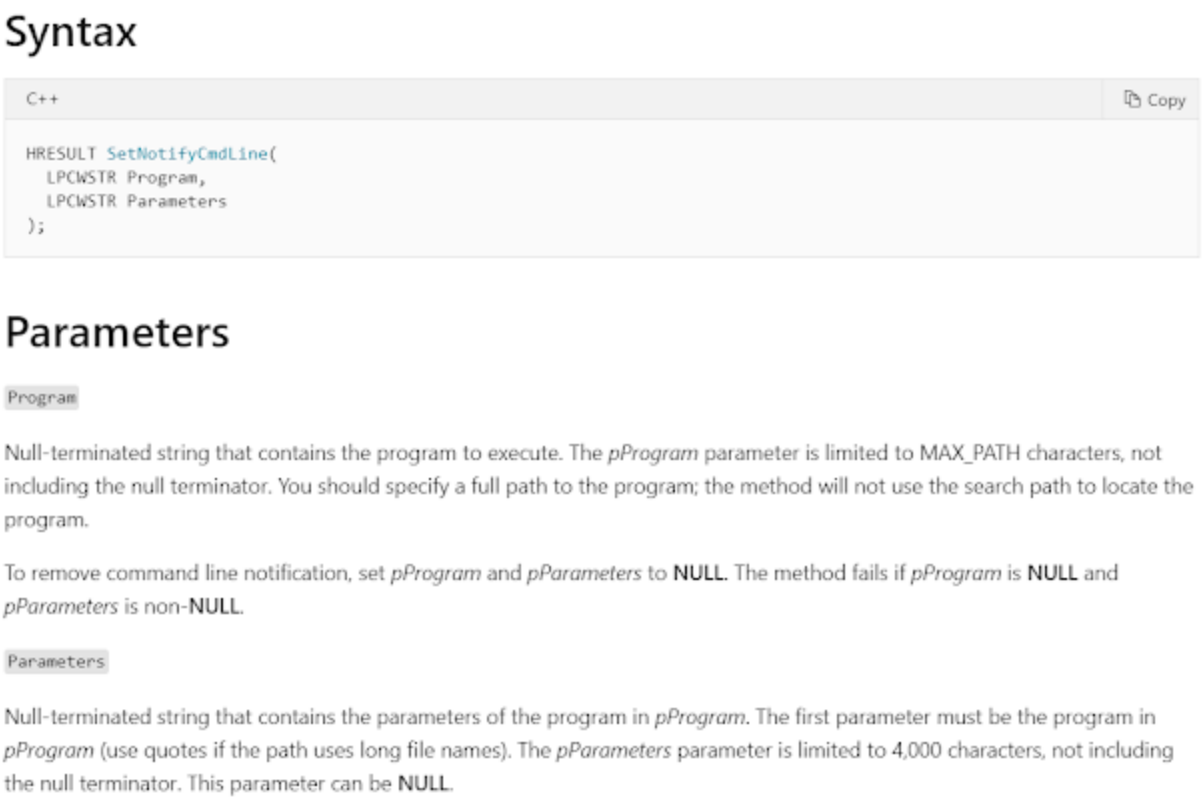


Name	Description
BG_JOB_TYPE_DOWNLOAD	Specifies that the job downloads files to the client.
BG_JOB_TYPE_UPLOAD	Specifies that the job uploads a file to the server. **BITS 1.2 and earlier:** not supported.
BG_JOB_TYPE_UPLOAD_REPLY	Specifies that the job uploads a file to the server, and receives a reply file from the server application. **BITS 1.2 and earlier:** not supported.

Figure 5 - BITS JOB TYPE

From a behavior perspective all the download and upload activities are performed by the BITS server (svchost.exe) impersonating the BITS client which breaks the link between the client and the server if using standard monitoring telemetry such as Sysmon network and file creation events.

BITS can be also abused for persistence by setting a command to run every time a JOB transfer job enters the BG_JOB_STATE_ERROR or BG_JOB_STATE_TRANSFERRED state using the IBackgroundCopyJob2::SetNotifyCmdLine method (i.e. bitsadmin.exe /SetNotifyCmdLine) which will result in a malicious program or command to be run persistently on a target system.



Syntax

C++ Copy

```
HRESULT SetNotifyCmdLine(
  LPCWSTR Program,
  LPCWSTR Parameters
);
```

Parameters

`Program`

Null-terminated string that contains the program to execute. The *pProgram* parameter is limited to MAX_PATH characters, not including the null terminator. You should specify a full path to the program; the method will not use the search path to locate the program.

To remove command line notification, set *pProgram* and *pParameters* to **NULL**. The method fails if *pProgram* is **NULL** and *pParameters* is non-**NULL**.

`Parameters`

Null-terminated string that contains the parameters of the program in *pProgram*. The first parameter must be the program in *pProgram* (use quotes if the path uses long file names). The *pParameters* parameter is limited to 4,000 characters, not including the null terminator. This parameter can be **NULL**.

Figure 6 - BITS SetNotifyCmdLine Parameters

Detection and Hunting

From a detection and forensics perspective Windows provides good logging events for the BITS client activities via the Microsoft-Windows-Bits-Client provider (enabled by default), key events are:

- EventID 3 - BITS service created a new Job
- EventID 59 - BITS started the <jobname> transfer job that is associated with http://example.com URL.
- EventID 60 - BITS stopped transferring the <jobname> transfer job that is associated with the http://example.com URL. The status code is 0xxxx.
- EventID 4 - The transfer job is complete
- EventID 5 - Job cancelled.
- Other events that are related to performance and transfer errors

Events such as 59, 60 and 61 contains the download/upload URL (very useful for forensics and detection) and event 3 contains the details of the BITS client process path and the Job name (very useful for detecting abnormal BITS clients).

Below example of BITS events resulting from this malware sample (Remcos or Netwire RAT loader):

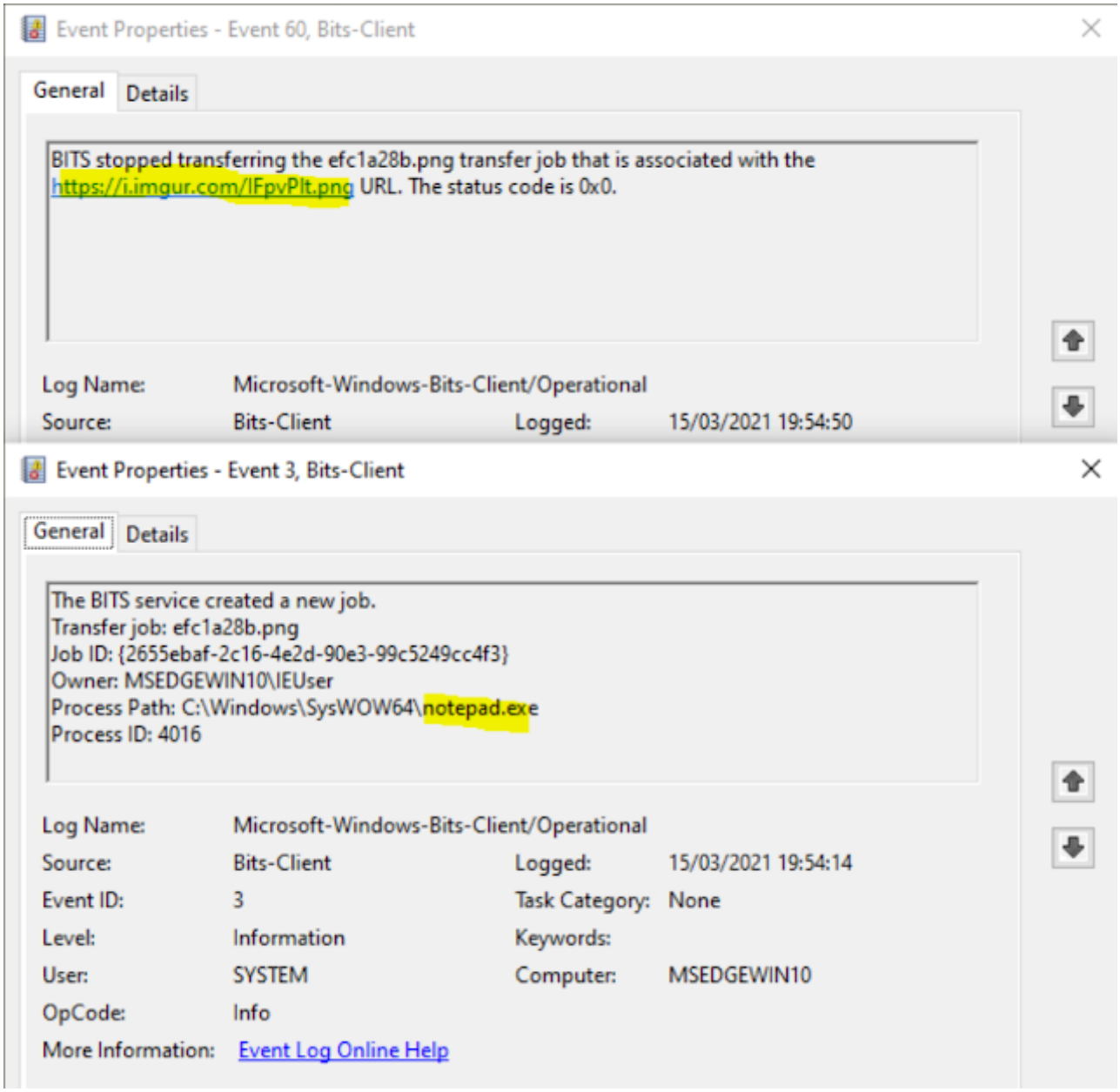


Figure 7 - BITS Client Event Logs 3 and 60

A) Unusual BITS Client:

By default on a Windows machine there are a limited number of BITS clients (native Windows binaries) and the majority are related to third party programs such as browsers. To baseline the clients we can use Bitsproxy.dll or qmgrprxy.dll ImageLoad events (such as Sysmon EventId 7) or BITS Client Event Logs EventId 3.

By default the following are the known normal BITS client with process path residing under c:\windows\ directory.

c:\Windows\SysWOW64\bitsadmin.exe
c:\Windows\System32\MDMAppInstaller.exe
c:\Windows\System32\DeviceEnroller.exe
c:\Windows\SysWOW64\OneDriveSetup.exe
c:\Windows\System32\ofdeploy.exe
c:\Windows\System32\directxdatabaseupdater.exe
c:\Windows\System32\MRT.exe (x)
c:\Windows\System32\aitstatic.exe
c:\Windows\System32\desktopimgdownldr.exe
c:\Windows\System32\Speech_OneCore\common\SpeechModelDownload.exe
c:\Windows\System32\RecoveryDrive.exe
c:\Windows\System32\svchost.exe (BITS service)

Excluding the above we can hunt/detect for unusual client, below an example of a hunting EQL query:

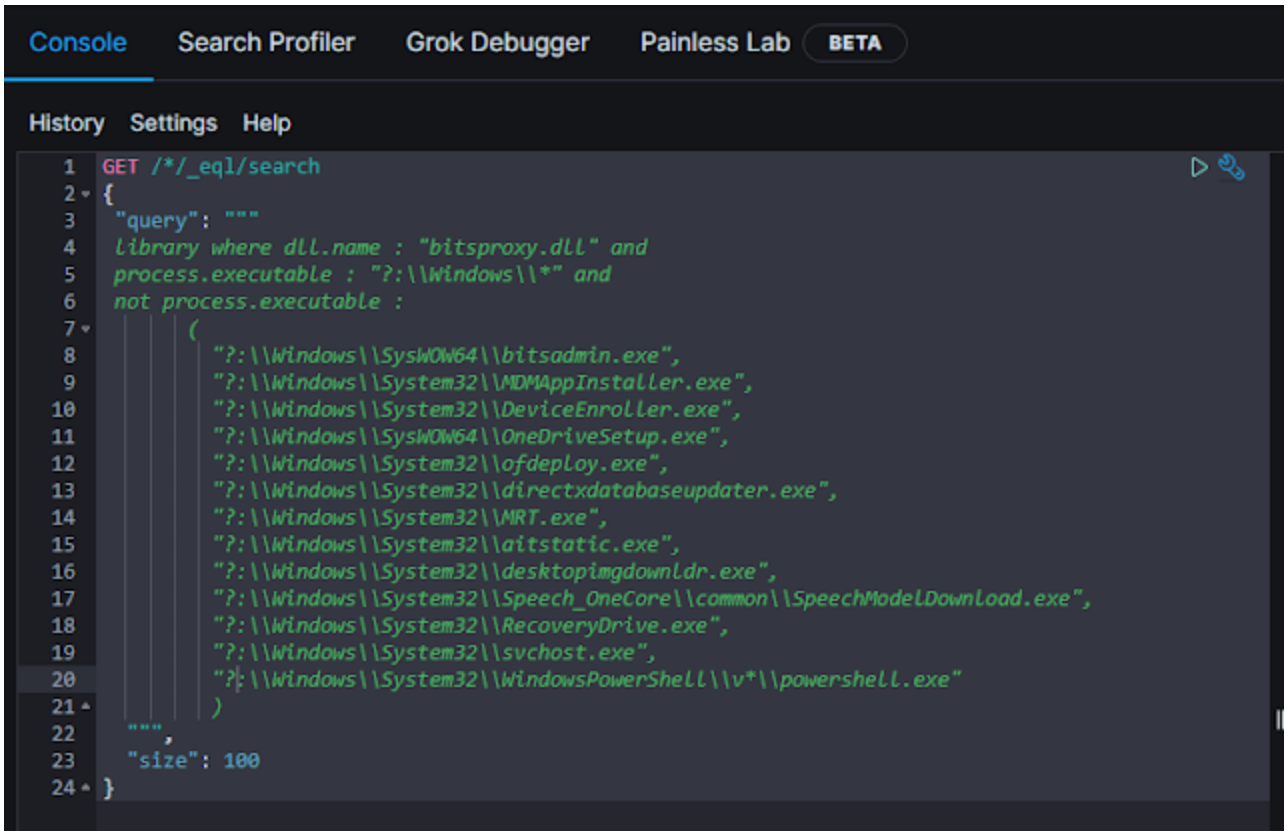


Figure 8 - Unusual Bits Client Hunt

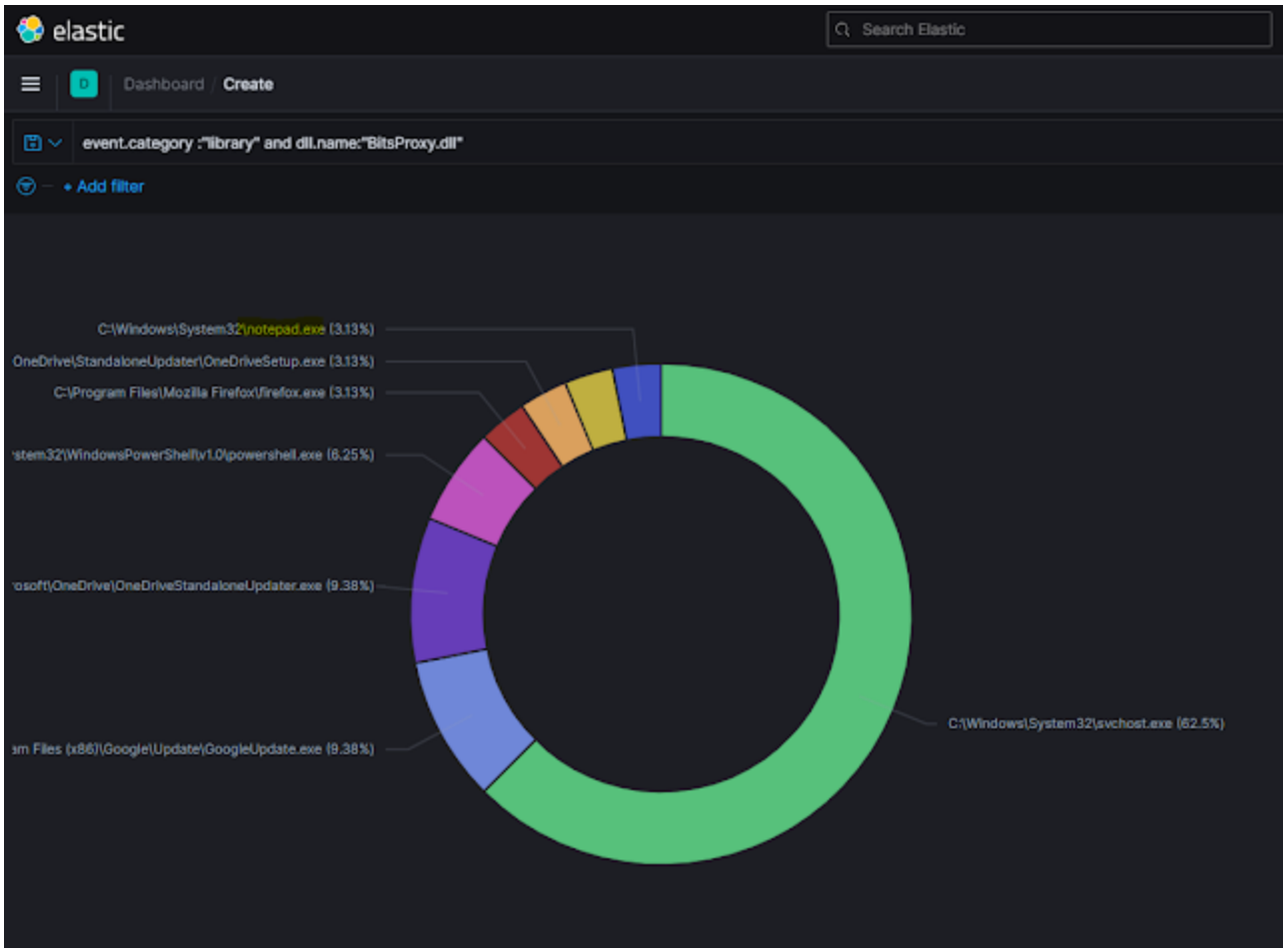


Figure 9 - Notepad.exe Unusual BITS Client

B) Program Execution via BITS SetNotifyCmdline Persistence:

Programs set to execute via the SetNotifyCmdline method are a child of the BITS service, there are some legit instances especially signed stuff running from program files directories, but its quite rare:

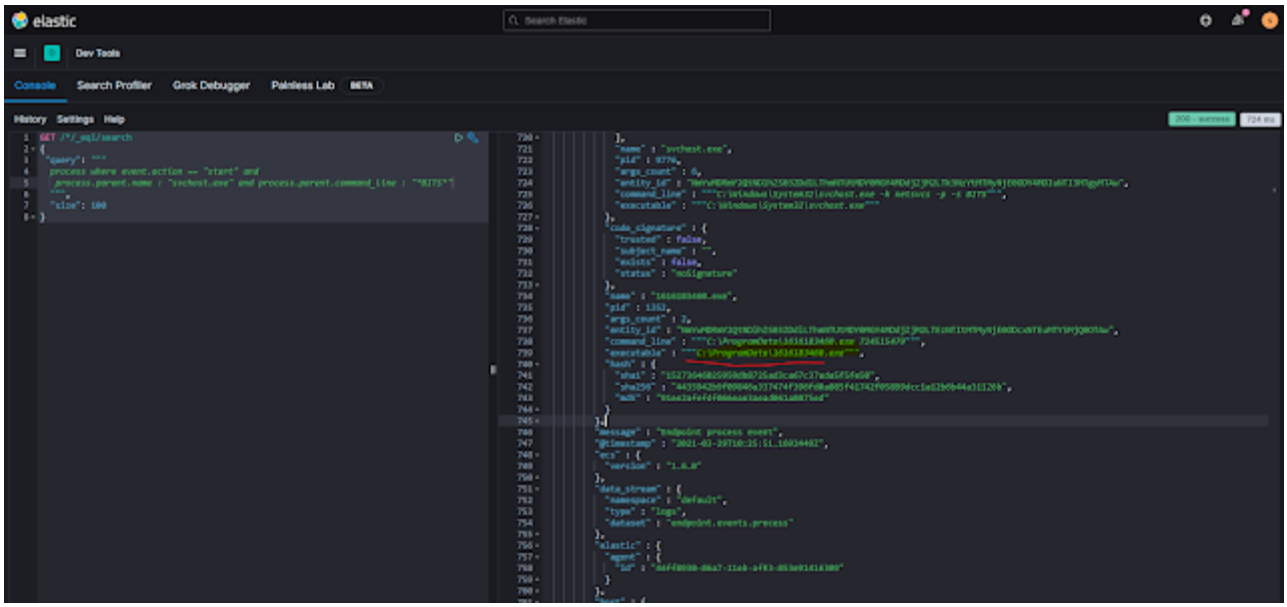


Figure 10 - Malware example persisting via BITS SetNotifyCmdline Method

C) Execution of a File Downloaded via BITS Service:

Last example is to hunt for executable content that is downloaded via BITS service and immediately executed, we can do that by correlating File rename event (old file name always follow this pattern BITXXXX.tmp and renamed to the target file name) by the BITS service followed by process execution by file.path/process.executable:

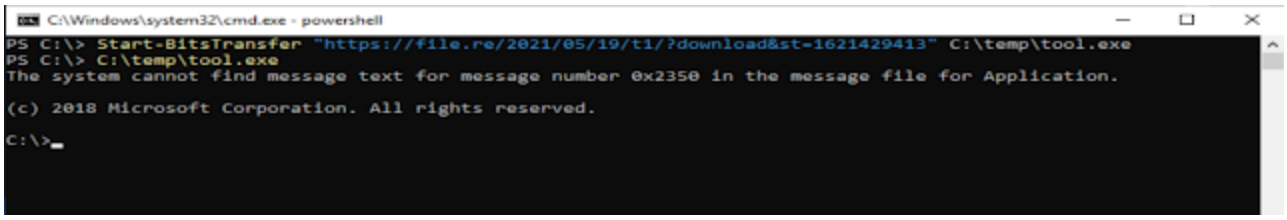


Figure 11 - Download & Execution of a file via BITS using PowerShell

As you can see below we can link the file download activity to the process execution event:

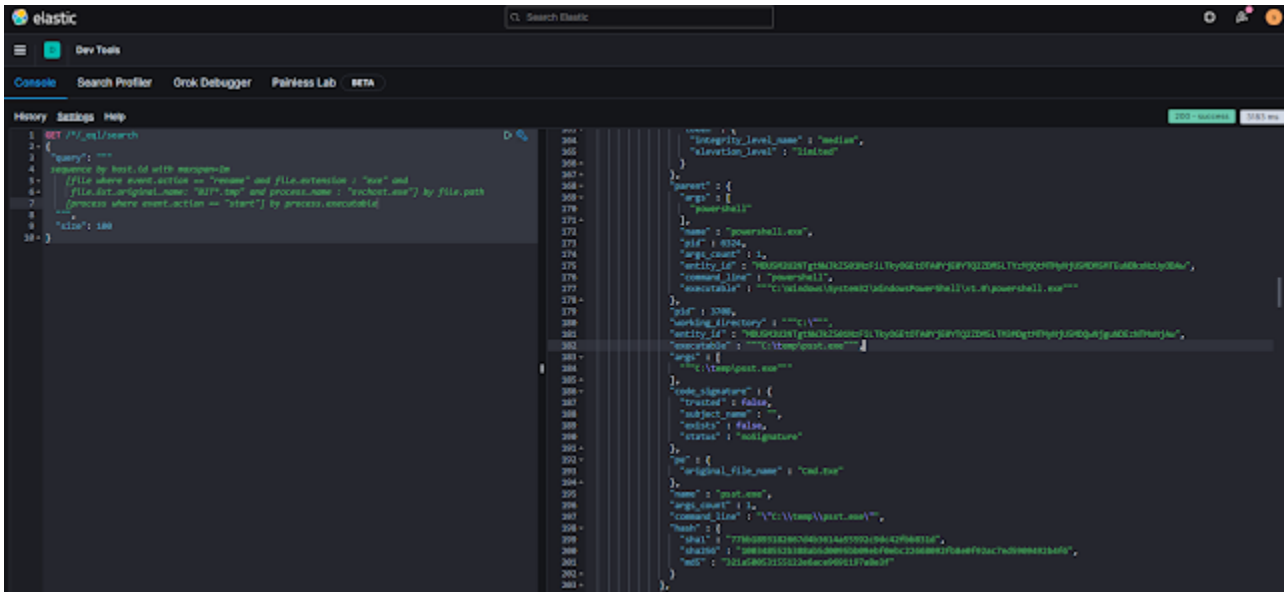


Figure 12 - Hunt for Process Execution of a File Downloaded via BITS service

There other scenarios but that should give you an idea of the building blocks and how to play with them to spot unusual combinations.