BITS and Bytes: Analyzing BITSLOTH, a newly identified backdoor

elastic.co/security-labs/bits-and-bytes-analyzing-bitsloth

Subscribe

Elastic Security Labs identified a novel Windows backdoor leveraging the Background Intelligent Transfer Service (BITS) for C2. This malware was found during a recent activity group tracked as REF8747.

19 min read

Malware analysis

BITSLOTH at a glance

BITSLOTH is a newly discovered Windows backdoor that leverages the Background Intelligent Transfer Service (BITS) as its command-and-control mechanism. BITSLOTH was uncovered during an intrusion within the LATAM region earlier this summer. This malware hasn't been publicly documented to our knowledge and while it's not clear who's behind the malware, it has been in development for several years based on tracking distinct versions uploaded to VirusTotal.

The most current iteration of the backdoor at the time of this publication has 35 handler functions including keylogging and screen capture capabilities. In addition, BITSLOTH contains many different features for discovery, enumeration, and command-line execution. Based on these capabilities, we assess this tool is designed for gathering data from victims.

Key takeaways

- BITSLOTH is a newly discovered Windows backdoor
- BITSLOTH uses a built-in Microsoft feature, Background Intelligent Transfer Service (BITS) for command-and-control communication
- BITSLOTH has numerous command handlers used for discovery/enumeration, execution, and collection purposes
- The backdoor contains logging functions and strings consistent with the authors being native Chinese speakers

Discovery

Our team observed BITSLOTH installed on a server environment on June 25th during REF8747, this was an intrusion into the Foreign Ministry of a South American government. The intrusion was traced back to PSEXEC execution on one of the infected endpoints. The attackers used a slew of publicly available tools for most of their operations with the exception of BITSLOTH.

One of the primary mechanisms of execution was through a shellcode loading project called RINGQ. In a similar fashion to DONUTLOADER, RINGQ will convert any Windows executable and generate custom shellcode placing it into a file (main.txt). This shellcode gets decrypted and executed in-memory. This technique is used bypass defenses that rely on hash blocklists or static signatures in some anti-malware products.

We observed RINGQ being used to load the IOX port forwarder. Note: The key in the image below is the hex conversion of “whoami”.

Additionally the attackers used the STOWAWAY utility to proxy encrypted traffic over HTTP to their C2 servers. Proxy tools, tunnelers, and redirectors are commonly used during intrusions to conceal the adversary responsible for an intrusion. These tools offer adversaries various features, including the ability to bypass internal network controls, provide terminal interfaces, encryption capabilities as well as file transfer options.

After initial access, the actor moved laterally and dropped BITSLOTH in the form of a DLL (`flengine.dll`) inside the ProgramData directory. The actor then executed the music-making program FL Studio (`fl.exe`). Based on the observed call stack associated with the self-injection alert, we confirmed the threat actor used a traditional side-loading technique using a signed version of FL Studio.

```
c:\windows\syswow64\ntdll.dll!0x770841AC
c:\windows\syswow64\ntdll.dll!0x7709D287
c:\windows\syswow64\kernelbase.dll!0x76ED435F
c:\windows\syswow64\kernelbase.dll!0x76ED42EF
Unbacked!0x14EAB23
Unbacked!0x14EA8B6
c:\programdata\pl studio\flengine.dll!0x74AD2F2E
c:\programdata\pl studio\fl.exe!0xDB3985
c:\programdata\pl studio\fl.exe!0xDB3E5E
c:\programdata\pl studio\fl.exe!0xDB4D3F
c:\windows\syswow64\kernel32.dll!0x76B267F9
c:\windows\syswow64\ntdll.dll!0x77077F4D
c:\windows\syswow64\ntdll.dll!0x77077F1B
```

This call stack was generated along with a process injection alert, and enabled researchers to extract an in-memory DLL that was set with Read/Write/Execute(RWX) page protections.

BITSLOTH overview

During our analysis, we found several older BITSLOTH samples demonstrating a record of development since December 2021. Within this project, the malware developer chose notable terminology– referring to BITSLOTH as the `Slaver` component and the command and control server as the `Master` component. Below is an example of one of the PDB file paths linked to BITSLOTH that depicts this:

BITSLOTH employs no obfuscation around control flow or any kind of string encryption.

Both older and recent samples contain strings used for logging and debugging purposes. As an example at startup, there is a string referenced in the read-only section (`.rdata`).

This Simplified Chinese wide-character string translates to: `Note: There is already a program running, do not run it again…`

These small snippets contained within BITSLOTH help shed light on the development and prioritization of features, along with what appear to be operator instructions. In the latest version, a new scheduling component was added by the developer to control specific times when BITSLOTH should operate in a victim environment. This is a feature we have observed in other modern malware families such as EAGERBEE.

BITSLOTH code analysis

BITSLOTH is a backdoor with many different capabilities including:

- Running and executing commands
- Uploading and downloading files
- Performing enumeration and discovery
- Collecting sensitive data through keylogging and screen capturing

Mutex

BITSLOTH uses a hard-coded mutex (`Global\d5ffff77ff77adad657658`) within each sample to ensure only one instance is running at a time.

Communication

BITSLOTH adopts a traditional client/server architecture, the developer refers to the client as the `Slaver` component and the command and control server (C2) as the `Master` component. The developer embeds the IP/port of the C2 server in each sample with a front-loaded string (`rrrr_url`). This string acts as a key to identify the C2 configuration in itself while running in memory, this is used when updating the C2 server.

Below are the configurations in several samples our team has observed, the threat actor configures both internal and external IP ranges.

```
rrrr_url216.238.121[.]132:8443
rrrr_url192.168.1[.]125:8443
rrrr_url192.168.1[.]124:8443
rrrr_url45.116.13[.]178:443
```

One of the defining features of BITSLOTH is using the Background Intelligent Transfer Service (BITS) for C2. While this feature has been designed to facilitate the network transfer of files between two machines, it's been abused by multiple state-sponsored groups and continues to fly under the radar against organizations. This medium is appealing to adversaries because many organizations still struggle to monitor BITS network traffic and detect unusual BITS jobs.

> Windows has a system administration feature called Background Intelligent Transfer Service (BITS) enabling the download and upload of files to HTTP web servers or SMB shares. The BITS service employs multiple features during the file transfer process such as the ability to pause/resume transfers, handling network interruptions, etc. BITS traffic is usually associated with software updates therefore wrongfully implied as trusted. Many organizations lack visibility into BITS network traffic making this an appealing target.

The BITS API is exposed through Window's Component Object Model (COM) using the **IBackgroundCopyManager** interface. This interface provides capabilities to create new jobs, enumerate existing jobs in the transfer queue, and access a specific job from a transfer queue.

After initialization, BITSLOTH cancels any existing BITS jobs on the victim machine that match the following display names:

- `WU Client Download`
- `WU Client Upload`
- `WU Client Upload R`

These names are used by the developer to blend in and associate the different BITS transfer jobs with their respective BITS job type. By canceling any existing jobs, this allows the execution of the malware to operate from a clean state.

Below are the Microsoft definitions matching the type of BITS job:

- **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.
- **BG_JOB_TYPE_UPLOAD_REPLY** - Specifies that the job uploads a file to the server, and receives a reply file from the server application.

After canceling any existing jobs, the MAC address and operating system information are retrieved and placed into global variables. A new thread gets created, configuring the auto-start functionality. Within this thread, a new BITS download job is created with the name (`Microsoft Windows`).

This download job sets the destination URL to `http://updater.microsoft[.]com/index.aspx`. While this domain is not routable, BITSLOTH masquerades this BITS job using a benign looking domain as a cover then uses **SetNotifyCmdLine** to execute the malware when the transfer state is changed.

Interestingly, this unique toolmark allowed us to pivot to additional samples showing this family has been in circulation for several years.

At this point, the malware has now been configured with persistence via a BITS job named `Microsoft Windows`. Below is a screenshot of this job's configuration showing the notification command line set to the BITSLOTH location (`C:\ProgramData\Media\setup_wm.exe`)

Once BITSLOTH becomes active, it will start requesting instructions from the C2 server using the `WU Client Download` job. This request URL is generated by combining the MAC address with a hard-coded string (`wu.htm`). Below is an example URL:

```
https://192.168.182.130/00-0C-29-0E-29-87/wu.htm
```

In response to this request, the malware will then receive a 12-byte structure from the C2 server containing a unique ID for the job, command ID for the handler, and a response token. Throughout these exchanges of file transfers, temporary files from the victim machine are used as placeholders to hold the data being transmitted back and forth, BITSLOTH uses a filename starting with characters (`wm`) appended by random characters.

Command functionality

BITSLOTH uses a command handler with 35 functions to process specific actions that should be taken on the victim machine. The malware has the option to be configured with HTTP or HTTPS and uses a hardcoded single byte XOR (`0x2`) to obfuscate the incoming instructions from the C2 server. The outbound requests containing the collected victim data have no additional protections by the malware itself and are sent in plaintext.

In order to move fast, our team leveraged a helpful Python implementation of a BITS server released by SafeBreach Labs. By setting the C2 IP to our loopback address inside a VM, this allowed us to get introspection on the network traffic.

The handlers all behave in a similar approach performing a primary function then writing the data returned from the handler to a local temporary file. These temporary files then get mapped to a BITS upload job called `WU Client Upload`. Each handler uses its own string formatting to create a unique destination URL. Each filename at the end of the URL uses a single letter to represent the type of data collected from the host, such as `P.bin` for processes or `S.bin` for services.

```
http://192.168.182.130/00-0C-29-0E-29-87/IF/P.bin
```

Below is an example screenshot showing the process enumeration handler with the string formatting and how this data is then linked to the BITS upload job.

This link to the exfiltrated data can also be observed by viewing the BITS upload job directly. In the screenshots below, we can see the destination URL (C2 server) for the upload and the temporary file (`wm9F0C.tmp`) linked to the job.

If we look at the temporary file, we can see the collected process information from the victim host.

Soon after the upload job is created, the data is sent over the network through a BITS_POST request containing the captured data.

Command handling table

Command ID	Description
0	Collect running processes via **WTSEnumerateProcessesW**
1	Get Windows services via **EnumServicesStatusW**
2	Get system information via `systeminfo` command
3	Retrieve all top-level Windows via **EnumWindows**
5	Collect file listings
6	Download file from C2 server
7	Upload file to C2 server
10	Terminate itself
11	Set communication mode to HTTPS
12	Set communication mode to HTTP
13	Remove persistence
14	Reconfigure persistence
15	Cancel BITS download job (`WU Client Download`)
16	Remove persistence and delete itself
17	Thread configuration
18	Duplicate of handler #2
19	Delete file based on file path
20	Delete folder based on file path
21	Starts terminal shell using stdin/stdout redirection
22	Resets terminal handler (#21)
23	Runs Windows tree command
24	Updates BITSLOTH, delete old version
25	Shutdown the machine via **ExitWindowsEx**
26	Reboot the machine via **ExitWindowsEx**
27	Log user off from the machine via **ExitWindowsEx**
28	Terminate process based on process identifier (PID)
29	Retrieves additional information via `msinfo32` command
30	Execute individual file via **ShellExecuteW**
34	Create new directory via **CreateDirectoryW**
41	Upload data to C2 server
42	Checks for capture driver via **capGetDriverDescriptionW**

Command ID	Description
43	Take screenshots of victim machine desktop
44	Record keystrokes from victim machine
45	Stop recording screenshot images
46	Stop keylogger functionality

Backdoor functionality

BITSLOTH includes a wide range of post-compromise capabilities for an adversary to operate within a victim environment. We will focus on the more significant capabilities by grouping them into different categories.

Discovery/enumeration

A portion of the BITSLOTH handlers are focused on retrieving and enumerating data from victim machines. This includes:

- Retrieving process information via **WTSEnumerateProcessesW**
- Collecting Windows services via **EnumServicesStatusW**
- Enumerating all top-level Windows via **EnumWindows** with a callback function
- Retrieving system information via windows utilities such as `systeminfo` and `msinfo32`

In many of the handlers, the locale version is configured to `chs` (Chinese - Simplified).

BITSLOTH has a couple custom enumeration functions tied to retrieving file listings and performing directory tree searches. The file listing handler takes a custom parameter from the operator to target specific folder locations of interest:

- **GET_DESKDOP** → **CSIDL_DESKTOPDIRECTORY** (Desktop)
- **GET_BITBUCKET** -> **CSIDL_BITBUCKET** (Recycle Bin)
- **GET_PERSONAI** -> **CSIDL_MYDOCUMENTS** (My Documents)

BITSLOTH also has the ability to collect entire directory/file listings on the machine for every file by using the Windows tree utility. This handler loops across the alphabet for each drive letter where the data is then saved locally in a temporary file named `aghzyxklg`.

The tree data is then compressed and sent to the C2 server with a .ZIP extension. Below is an example of the collected data. This data can help pinpoint sensitive files or provide more context about the target environment.

Collection

In terms of collection, there are a few handlers used for actively gathering information. These are centered around capturing screenshots from the desktop and performing keylogging functionality.

BITSLOTH implements a lightweight function used to identify capture recording devices, this appears to be a technique to check for a camera using the Windows API (**capGetDriverDescriptionW**).

BITSLOTH has the ability to take screenshots based on parameters provided by the operator. Input to this function uses a separator (`||`) where the operator provides the number of seconds of the capture interval and the capture count. The images are stored as BMP files with a hard coded name `ciakfjoab` and compressed with the DEFLATE algorithm using a `.ZIP` archive. These timestamped zipped archives are then sent out to the C2 server.

The handler leverages common screenshot APIs such as **CreateCompatibleBitmap** and **BitBlt** from `Gdi32.dll`.

For recording keystrokes, BITSLOTH uses traditional techniques by monitoring key presses using **GetAsyncKeyState**/**GetKeyState**. The handler has an argument for the number of seconds to perform the keylogging. This data is also compressed in a `.ZIP` file and sent outbound to the C2 server.

Execution / Maintenance

BITSLOTH has multiple capabilities around maintenace and file execution as well as standard backdoor functionalities such as:

- Capability to execute files stand-alone via **ShellExecuteW**
- Windows terminal capability to execute commands and read data back via pipes
- Create directories, perform reboots, shutdown the machine, terminate processes
- Perform file upload and download between C2 server
- Modify BITSLOTH configuration such as communication modes, update C2 URL, turn off keylogging/screenshot features

BITSLOTH pivots

BITSLOTH appears to be actively deployed. We identified another BITSLOTH C2 server (`15.235.132[.]67`) using the same port (`8443`) with the same SSL certificate used from our intrusion.

While it’s not exactly clear who’s behind BITSLOTH, there was a large amount of activity of VirusTotal uploads occurring on December 12, 2021. With around 67 uploads over 24 hours from one submitter (`1fcc35ea`), we suspect someone linked to this project was validating detections, making modifications, and uploading different versions of BITSLOTH to VirusTotal. One sample was packed with VMProtect, others stripped of functionality, some uploads were debug builds, etc.

A lot of time has passed since then, but it is interesting seeing this family show up in a recent intrusion. Whatever the objective behind this malware, it's surprising that this family remained under the radar for so many years.

REF 8747 through MITRE ATT&CK

Elastic uses the MITRE ATT&CK framework to document common tactics, techniques, and procedures that advanced persistent threats use against enterprise networks.

[h4] Tactics Tactics represent the why of a technique or sub-technique. It is the adversary’s tactical goal: the reason for performing an action.

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

Detecting REF8747

Detection

The following detection rules and behavior prevention events were observed throughout the analysis of this intrusion set:

YARA Signatures

YARA

Elastic Security has created YARA rules to identify this activity. Below are YARA rules to identify BITSLOTH:

```
rule Windows_Trojan_BITSLOTH_05fc3a0a {
    meta:
        author = "Elastic Security"
        creation_date = "2024-07-16"
        last_modified = "2024-07-18"
        os = "Windows"
        arch = "x86"
        threat_name = "Windows.Trojan.BITSLOTH"
         license = "Elastic License v2"

    strings:
        $str_1 = "/%s/index.htm?RspID=%d" wide fullword
        $str_2 = "/%s/%08x.rpl" wide fullword
        $str_3 = "/%s/wu.htm" wide fullword
        $str_4 = "GET_DESKDOP" wide fullword
        $str_5 = "http://updater.microsoft.com/index.aspx" wide fullword
        $str_6 = "[U] update error..." wide fullword
        $str_7 = "RMC_KERNEL ..." wide fullword
        $seq_global_protocol_check = { 81 3D ?? ?? ?? ?? F9 03 00 00 B9 AC 0F 00 00 0F 46 C1 }
        $seq_exit_windows = { 59 85 C0 0F 84 ?? ?? ?? ?? E9 ?? ?? ?? ?? 6A 02 EB ?? 56 EB }
    condition:
        2 of them
}
```

Observations

All observables are also available for download in both ECS and STIX format in a combined zip bundle.

The following observables were discussed in this research.

Observable	Type	Name	Reference
4a4356faad620bf12ff53bcfac62e12eb67783bd22e66bf00a19a4c404bf45df	SHA-256	`s.dll`	BITSLOTH
dfb76bcf5a3e29225559ebbdae8bdd24f69262492eca2f99f7a9525628006d88	SHA-256	`125.exe`	BITSLOTH
4fb6dd11e723209d12b2d503a9fcf94d8fed6084aceca390ac0b7e7da1874f50	SHA-256	`setup_wm.exe`	BITSLOTH
0944b17a4330e1c97600f62717d6bae7e4a4260604043f2390a14c8d76ef1507	SHA-256	`1242.exe`	BITSLOTH
0f9c0d9b77678d7360e492e00a7fa00af9b78331dc926b0747b07299b4e64afd	SHA-256	`setup_wm.exe`	BITSLOTH (VMProtect)
216.238.121[.]132	ipv4-addr	BITSLOTH C2 server	
45.116.13[.]178	ipv4-addr	BITSLOTH C2 server	
15.235.132[.]67	ipv4-addr	BITSLOTH C2 server	
http ://updater.microsoft.com/index.aspx			BITSLOTH file indicator
updater.microsoft.com			BITSLOTH file indicator

References

The following were referenced throughout the above research:

About Elastic Security Labs

Elastic Security Labs is the threat intelligence branch of Elastic Security dedicated to creating positive change in the threat landscape. Elastic Security Labs provides publicly available research on emerging threats with an analysis of strategic, operational, and tactical adversary objectives, then integrates that research with the built-in detection and response capabilities of Elastic Security.

Follow Elastic Security Labs on Twitter @elasticseclabs and check out our research at www.elastic.co/security-labs/.