Hiding in Plain Sight: The Subtle Art of Loki Malware's Obfuscation

logpoint.com/en/blog/hiding-in-plain-sight-the-subtle-art-of-loki-malwares-obfuscation/

Anish Bogati November 7, 2024

With the surge of cyberattacks, sharing threat intelligence in the form of insights, trends, and samples is crucial to combat new and old threats effectively. Independent security researchers worldwide contribute to different repositories, which play a key role in enabling other security researchers and analysts to develop detection rules and response tactics to stay ahead of emerging threats. One of them to be aware is the malware named Loki.

Anish Bogati
Global Services and Security Research

Background

While browsing through recent uploads in MalwareBazaar — a comprehensive database of known malware samples — we discovered a Loki malware sample belonging to a previously unexamined malware family.

Loki is a type of information-stealing malware known for exfiltrating sensitive data, such as credentials, cryptocurrency wallets, and other personal information, often targeting Windows systems. It typically employs various techniques for persistence, obfuscation, and communication with its command-and-control (C2) servers, making it a significant threat in the cyber landscape.

Intrigued by its potential uniqueness, we selected it for further analysis. In this blog, we will focus exclusively on the initial stages of the infection.

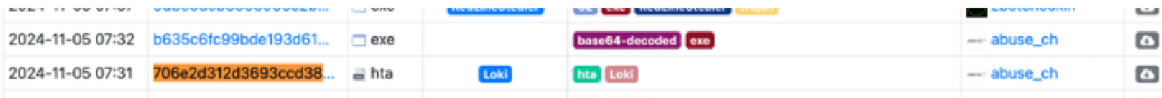
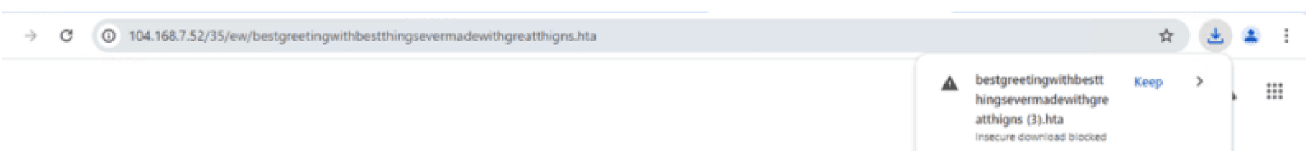


image-20241106-083707
MalwareBazaar Sample

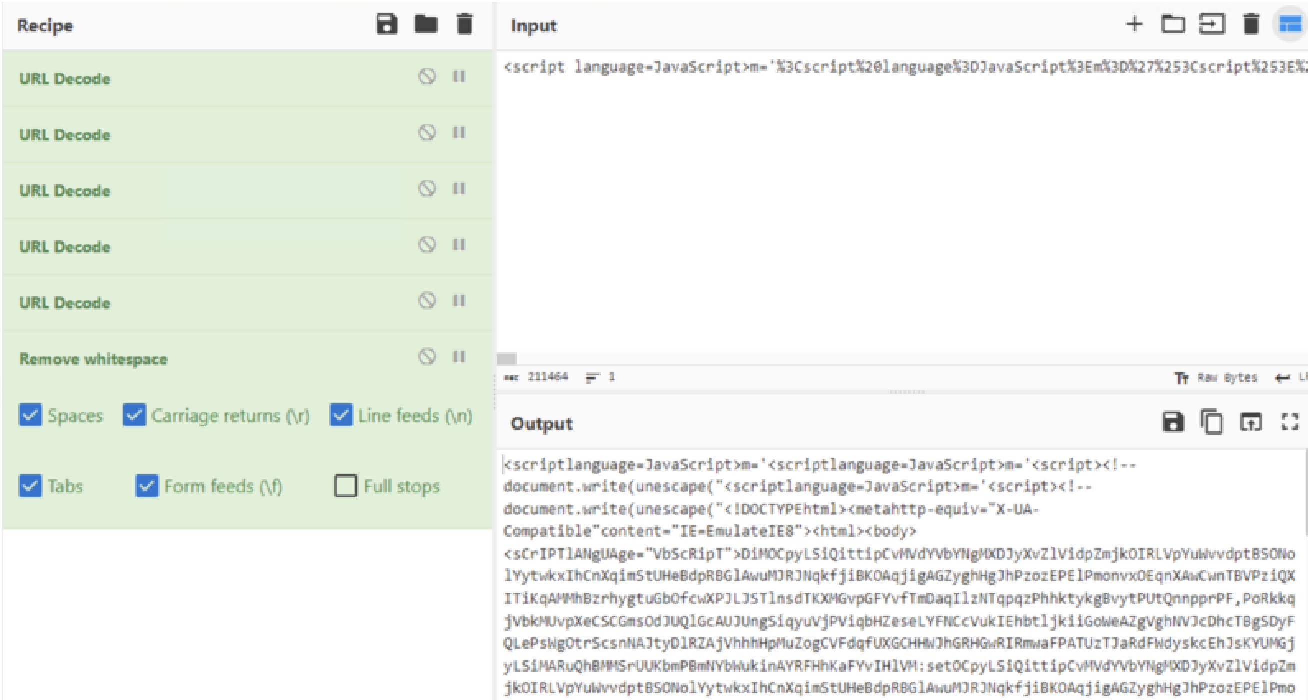
Sample Analysis

During dynamic analysis, the sample exhibited several familiar behaviors often observed in other malware we've encountered regularly. However, as we dug deeper, we noticed a range of underlying functions that set it apart. Interestingly, at the time of this analysis, the initial delivery sites for this malware were still active.



Downloading the payload from the site

With the sample downloaded, let's dive into the analysis. The initial HTA file contains multiple layers of URL encoding. After decoding, we found the payload was further obfuscated using Base64 encoding and a character substitution technique.



Decoding the HTA payload in CyberChef

We obtained a PowerShell command that, once decoded, revealed the following.

This is the decoded code:

The payload uses PowerShell to execute additional actions. Specifically, it loads `urlmon.dll` and leverages its functions to download a payload from the URL `hxxp://104[.]168[.]7[.]52/35/picturewithattitudeeventforallthings.tif`. Once downloaded, this file is saved as `picturewithattitudeeventforallthings.vbs` under `%user%\AppData\Roaming\` directory.

After the VBS file was executed with `wscript.exe`, the following command was executed:

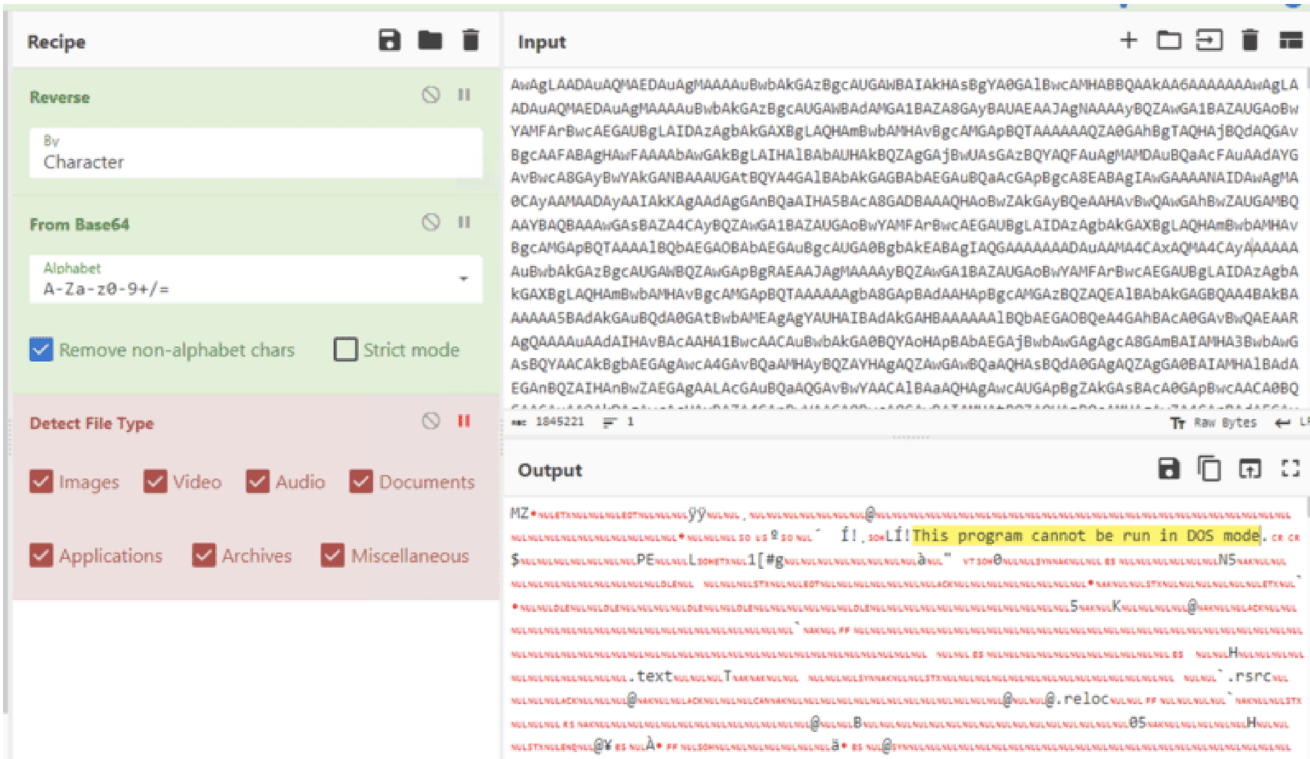
Once again, Base64 encoding and junk character insertion were used to obfuscate the command. The purpose of this command is to download an image from Google Drive at the URL `hxxps://drive[.]google[.]com/uc?export=download&id=1UyHqwrnXClKBJ3j63Ll1t2StVgGxbSt0`.

Steganography has been applied to the image to conceal additional Base64-encoded instructions.

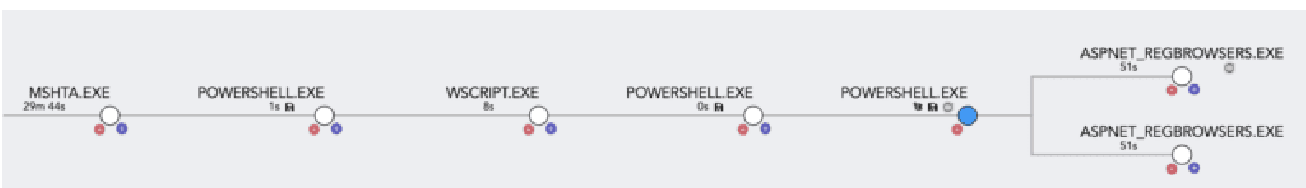
Google Drive hosted image

The script retrieves the hidden obfuscated, reversed Base64 payload from the Google Drive image, decodes it, loads it as a .NET assembly, and then invokes a method within that assembly. Each step includes layers of obfuscation—such as string concatenation, junk insertion, Base64 reversal, and dynamic replacements—to hinder analysis and evade detection. This technique, commonly used in malware, allows malicious code to be loaded dynamically without being directly written to disk.

Here is the Base64-encoded payload, which was embedded in reverse order within the image.

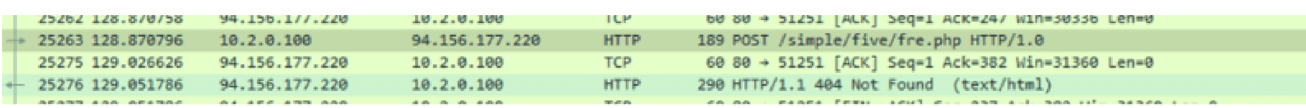


In summary, the VBS payload instructs the system to visit an image hosted on Google Drive, where it retrieves a hidden, Base64-encoded payload. This encoded portion is then reversed, decoded, and the code is injected into the `aspnet_regbrowsers.exe` process as seen in the Process Tree below.



Process tree of infection chain

Then the injected process further starts to communicate with C2, and attempts to drop other payloads into the system which at the moment of testing was already down so were not able to observe further activities.



Network Connection to C2

Detection of Loki with Logpoint SIEM

As demonstrated in the Loki sample analyzed above, the techniques employed are commonly utilized by other initial loaders and droppers to evade detection. Detecting these techniques is critical, as they reflect an increasing trend among malware to bypass conventional defenses.

To effectively detect these behaviors, having proper auditing configurations in place is crucial to ensure the generation of relevant logs. Specific log sources are fundamental for effective threat detection and hunting. Below is a list of key sources required for our detection strategy with Logpoint SIEM:

1. **Windows**

    Process creation with command-line auditing should be enabled.

2. **Windows Sysmon**

    To get started, you can use our sysmon baseline configuration.

3. **Network Logs**

    Firewall, IDS/IPS logs

Below is a list of vendor alerts that can help detect the aforementioned techniques used by malware.

1. **Suspicious MSHTA Process Pattern**

The initial payload execution of `.vbs` was done with `mshta.exe` a Windows internal binary. This alert can detect such behavior as it looks for the execution of `mshta.exe` from suspicious locations or the execution of file from a non-standard path.

2. Suspicious PowerShell Parameter Substring Detected
Given that many of the attack steps utilized PowerShell and its cmdlets, this alert detects the use of suspicious PowerShell commandlets commonly linked to malicious activities, such as executing Base64-encoded payloads or downloading remote files through PowerShell cmdlets.

3. Usage of Web Request Command

Multiple stages of payloads were downloaded, so this alert can be used to detect such events where Windows binary and powershell commandlets have been used to download files.

4. Suspicious File Execution Using Wscript or Cscript

The VBS payload was executed using `wscript.exe`, making this alert effective for detecting the execution of scripting files such `vbs` files via wscript.exe or cscript.exe.

Note: *Alerts may generate false positives, so it's important to thoroughly test them within your environment before deploying them broadly. Conducting tests will help identify and filter out any false positives, as certain applications or specific legitimate use cases could trigger these alerts inadvertently.*

Recommendations

- **Block Execution of Suspicious File Types and Windows Binaries:**
  Block potentially exploited file types such as `.vbs`, `.hta` , and `.msi` , which are commonly used by threat actors for payload distribution. Allow exceptions only for trusted system processes or specific users to avoid disrupting legitimate use cases.

- **Restrict User Permissions and Software Installation:**
  Limit users' ability to install and run unauthorized software.

- **Regular Software Updates:**
  Ensure devices, browsers, and other software applications are regularly updated to protect against known vulnerabilities and cyber threats.

- **Implement Endpoint Detection and Response (EDR) Solutions:**
  Deploy advanced EDR tools to monitor suspicious activity, especially around script execution and binary downloads. This helps detect malware behaviors early, particularly when unconventional techniques, like those seen in the Loki malware analysis, are used.

- **Monitor and Restrict Web Browsing:**
  Monitor users' web browsing habits and restrict access to potentially harmful websites or content that could lead to malware downloads.

- **Enhance System Monitoring and Logging:**
  Proper logging, asset visibility, and system monitoring are critical for cybersecurity. Implement regular auditing to track user activity and identify anomalies. Comprehensive log collection across all systems is essential for effective threat detection and analysis

- **Ensure Proper Log Retention and Visibility:**
  Establish a log retention policy to store system and network logs for at least six months. This will provide sufficient data to trace the origin and timeline of any security incident, ensuring a comprehensive response.