Analysis Report for: 242417BCB787A1D673F9D63B8345BA9D.exe

Overall Functionality

This C code is an obfuscated piece of malware designed to download and execute malicious code. It leverages a combination of JavaScript and VBScript within an HTML wrapper to achieve this. The core functionality is hidden within Base64 encoded strings and reversed strings, making analysis challenging. The code first downloads a malicious payload from a remote server, and then executes it using ActiveX objects and VBScript's `CreateObject` function. The use of multiple scripting languages and encoding techniques aims to evade detection by security software. The downloaded payload's filename suggests it's likely some type of backdoor or remote access trojan. The code also attempts to hide itself by moving the browser window off-screen.

Function Summaries

* **`haveMy(loveFriend)` (JavaScript):** Creates an ActiveXObject with the given name (`loveFriend`). Returns the created ActiveXObject. This is used to interact with the operating system.

* **`andIFriend(iHaveAnd)` (JavaScript):** Gets the innerHTML of the element with the ID specified by `iHaveAnd`. Returns the innerHTML as a string.

* **`uU()` (JavaScript):** A helper function that retrieves the content of the HTML element with ID 'girlMy'. This element contains a Base64 alphabet string. Returns a string.

* **`girlIU(s)` (JavaScript):** This function performs Base64 decoding. It takes a Base64 encoded string (`s`) as input, utilizes the Base64 alphabet retrieved from `uU()`, and decodes it. It then returns the decoded string.

* **`boyFriendU(boysMy)` (JavaScript):** Reverses a string. Takes a string (`boysMy`) as input, reverses it, and returns the reversed string. This is used for obfuscation.

* **`loveLove(boysMy)` (JavaScript):** A wrapper function that calls `girlIU` after reversing its input string using `boyFriendU`. It adds additional obfuscation. This is also used for decoding.

* **`girlMyYou(s)` (JavaScript):** This function takes a string `s` containing JavaScript code and executes it using a dynamically created function.

* **`boyIBoy(girlsGirl)` (VBScript):** Creates a JavaScript engine using ActiveXObject. It sets language to "javascript" and a timeout, then uses `girlMyYou` to execute the javascript provided to it, effectively acting as a JavaScript interpreter within VBScript.

Control Flow

The main control flow is as follows:

1. The HTML embeds JavaScript and VBScript code.
2. JavaScript functions `loveLove` and helper functions decode Base64-encoded strings contained within the HTML's `

` elements.
3. The decoded strings represent JavaScript code and a URL to download a malicious file.
4. An ActiveXObject (`ActiveXObject("msxml2.xmlhttp")`) is used to download a file from the URL.
5. The downloaded payload's content is written to a file.
6. VBScript's `boyIBoy` function acts as a bridge, executing JavaScript code (received from the decoding process) using `girlMyYou`. This likely contains instructions to execute the downloaded payload.
7. The browser window is resized and moved off-screen to conceal the activity.

Data Structures

* **Strings:** The primary data structure used is strings, which hold encoded malicious code, URLs, and decoded instructions.
* **ActiveX Objects:** ActiveXObjects are created and used to download files and execute JavaScript code, representing a crucial part of the malware's functionality.
* **JavaScript Objects:** The Javascript code uses objects such as the ones created by `haveMy()` to interact with the system and manipulate the environment.

Malware Family Suggestion

Based on the code's functionality, this malware strongly resembles a **downloader/dropper** and possibly a backdoor or remote access trojan (RAT). The downloader retrieves a malicious payload from a remote server, and the subsequently executed code implies further malicious actions on the infected system. The use of obfuscation and multiple scripting languages is a common characteristic of advanced malware families that try to bypass detection mechanisms. The filename "haveSimpleAnd.jpg" appears to be a decoy, intended to hide the actual nature of the downloaded payload. Without analyzing the downloaded payload, a precise malware family classification is not possible. However, the techniques employed strongly suggest sophistication and malicious intent.