Analysis Report for: 2FF04B29E9D83643C01AD22E16C88A3F.exe

Overall Functionality

This C code is heavily obfuscated. Its primary purpose is to decode and execute a series of operations on a hexadecimal string provided as input (`ynbozcenhyx`). The obfuscation techniques include:

* **Encoded Strings:** Variable and function names, as well as string literals, are heavily obfuscated using seemingly random character sequences.
* **Array Indexing:** Complex array indexing is used to extract characters from strings and construct other strings.
* **Mathematical Operations:** Bitwise XOR operations are performed on parts of the decoded data.
* **Base64 Encoding (Likely):** The code's structure suggests base64 encoding is used somewhere for additional encoding.

The decoded data is likely machine code or instructions for a specific malicious action. The code's structure strongly suggests a malicious intent, possibly a downloader or a component of a larger malware operation. It avoids using standard C libraries extensively to evade detection.

Function Summaries

* **`main` (implicitly defined):** The entry point of the program. It takes a hexadecimal string as input and passes it to the `jlaadcfedet` function. It has no explicit return value, but implicitly returns 0 upon successful execution.

* **`jlaadcfedet`:** This function decodes the input hexadecimal string (`ynbozcenhyx`) into an array of integers (`feldbo`), likely representing bytecode or other data. It processes the input in 2-byte chunks, converts them to integers (base 16), and stores them in the `feldbo` array. It then calls the `kjoyzkixyw4` function. It takes the input string as a parameter and has no return value.

* **`ijkdyovlla`:** This function iterates through the `feldbo` array and applies some transformation to each element using the `THQa5f` string and a complex function call. This likely involves further decoding or manipulation of the data. The function has no parameters and no return value.

* **`bakakxak`:** This function joins the elements of the `feldbo` array into a single string (`zcuhaqox`) using an empty string as a separator and then calls the `awvxodbe` function. It has no parameters and no return value.

* **`awvxodbe`:** This function assigns the `zcuhaqox` string to a key in the `tysdyjogsup` structure, potentially as a result of a second layer of decoding or encryption. It has no parameters and no return value.

* **`kjoyzkixyw4`:** This function iterates through a range determined by the length of `akloq0` and XORs elements of the `feldbo` array with elements of the `qozuwzcigqi` array. `qozuwzcigqi` is generated from `akloq0` in a way that suggests a simple XOR cipher for additional obfuscation. The function has no parameters and no return value.

Control Flow

* **`main`:** Simple linear flow: calls `jlaadcfedet`, then `tysdyjogsup[...]`.

* **`jlaadcfedet`:** A `while` loop iterates through the input string, processing it in 2-byte chunks. The loop terminates when all chunks are processed. It then calls `kjoyzkixyw4`.

* **`ijkdyovlla`:** A `for` loop iterates through the `feldbo` array, applying a transformation to each element.

* **`kjoyzkixyw4`:** Contains nested loops. The outer loop iterates based on the length of `akloq0`, the inner loop iterates through `feldbo`, performing bitwise XOR operations.

Data Structures

* **`feldbo`:** An array of strings that is used to store the decoded bytes from the input hexadecimal string. Later, these are treated as integers.

* **`tysdyjogsup`:** A structure (likely a dictionary or hash table implemented as an array) that stores the final decoded result.

* **`qozuwzcigqi`:** An array that is used in the XOR operation within the `kjoyzkixyw4` function.

Malware Family Suggestion

Given the heavy obfuscation, the use of XOR operations, the structure of decoding from a hex string to an array of integers (likely bytes), and the final storage in a dictionary-like structure, this code strongly resembles a **downloader** or a component of a polymorphic **malware**. The exact family is impossible to determine without further analysis of the decoded payload. The obfuscation is designed to hinder static analysis and antivirus detection. The code avoids using standard library functions, hinting at an attempt to be stealthy. The techniques used are commonly seen in malware samples to evade detection.