Analysis Report for: Copy_MacroVariables_To_Server_V4.pyc

Decoded using latin-1...

The provided C code is heavily obfuscated and contains many non-printable characters. A precise analysis is impossible without significant deobfuscation. However, based on the discernible fragments and patterns, we can make some educated guesses.

Overall Functionality

The code appears to be a malicious program designed to interact with an FTP server. The presence of FTP-related function names (like `retrbinary`), file paths, and IP addresses strongly suggest this. It seems to download files from a remote FTP server, possibly containing malicious code, and then executes or uses them locally. The code also includes logging functionality, possibly to track its actions. There's evidence of interactions with tool data and possibly CNC machine control (mentions of `CNCPROGR`, tool descriptions, etc.). This strongly suggests an industrial control system (ICS) attack vector.

Function Summaries

Due to obfuscation, precise function summaries are difficult. We can only offer partial inferences based on visible function names and context:

* **`retrbinary`:** (Likely a wrapper around a standard FTP library function.) Downloads a binary file from the FTP server. Parameters would include the FTP connection, file path on the server, and potentially a local file path for saving the downloaded data.
* **`write`:** Writes data to a file.
* **`quit`:** (Likely closes the FTP connection).
* **`makedirs`:** Creates directories if they don't exist.
* **`open`:** Opens a file.
* **`copy2`:** Copies files.
* **`replace`:** Replaces text within a file or string.
* **`split`:** Splits strings.
* **`search`:** (Likely uses regular expressions) Searches for patterns in strings.
* **`format`:** Formats strings.
* **`int`:** (Likely converts string to integer).
* **`group`:** (Likely from regular expression matching) Extracts captured groups.
* **`readlines`:** Reads lines from a file.
* **`is_identical`:** Compares two files or strings for equality.
* **`strftime`:** Formats time.
* **`remove`:** Deletes a file.
* **`upper`:** Converts a string to uppercase.
* `find_product_name`: likely searches for product names.

Control Flow

The control flow is extremely difficult to analyze due to the obfuscation. There are likely conditional statements based on file existence, FTP success/failure, and potentially other factors controlling the execution path of the malware.

Data Structures

No explicit data structures are clearly defined, but the code likely uses strings (for file paths, IP addresses, etc.), integers (for counters, IDs), and possibly arrays or structures to store collected data. The extensive use of non-printable characters and string manipulation hints at dynamic data structure generation or usage.

Malware Family Suggestion

Given the code's FTP interaction, file manipulation, and potential CNC machine targeting, it is strongly suggestive of an **ICS malware** designed for espionage or sabotage. The obfuscation points towards advanced persistent threats (APTs) or targeted attacks. The specific malware family cannot be identified definitively from the provided snippet. However, the sophistication of the obfuscation suggests that this isn't a simple, readily-available malware tool, but rather a custom-built payload.

Critical Issues

* **Obfuscation:** The extreme obfuscation makes reverse engineering exceptionally difficult and time-consuming. This is a common tactic used by advanced malware.
* **FTP Communication:** The code directly interacts with an FTP server, making it vulnerable to network-based detection and analysis.
* **File System Manipulation:** The code creates, writes, reads, and potentially deletes files on the local system. This increases the potential for damage and leaves traces of activity.
* **CNC Interaction (suspected):** The evidence of interaction with CNC machines and tool data raises serious concerns about potential industrial sabotage.

To further analyze this code, extensive deobfuscation and careful dynamic analysis in a sandboxed environment would be necessary. This analysis should be conducted by a security professional with experience in malware reverse engineering. The use of debuggers and disassemblers would be crucial to understand the exact behavior of the code.