Analysis Report for: ED9D0EA96E56E94F2D1941486AD2BBAA.exe

Overall Functionality

This Python code creates a class `Images` that loads and processes several images encoded in Base64 format. The images are stored as class attributes, and their names are defined in the `_names` list. Each image is loaded from a Base64 string, converted to a bitmap using the `wxPython` library, and then assigned as an attribute to the `Images` class instance. The code uses `eval()` which is a significant security risk.

Function Summaries

* **`__init__(self)`:** This is the constructor for the `Images` class. It takes no parameters other than `self`. It iterates through the list `_names`, decodes each Base64-encoded image string (using `a2b_base64`), loads the image into a `wx.Image` object, converts its alpha channel to a mask, converts it to a bitmap, and assigns the bitmap to an attribute of the `Images` object with the same name as the string in the list. It also assigns the _names list to a class attribute.

Control Flow

The `__init__` function's control flow is straightforward:

1. **Iteration:** A `for` loop iterates through each element (`name`) in the `_names` list.
2. **Base64 Decoding:** Inside the loop, `eval(name)` is used to evaluate a string containing the Base64 image data which is highly insecure and dangerous. This is then decoded using `a2b_base64`.
3. **Image Loading:** The decoded data is passed to `wx.ImageFromStream(StringIO(...))` to create a `wx.Image` object. This function reads data from a stream.
4. **Alpha to Mask Conversion:** `img.ConvertAlphaToMask(100)` converts the image's alpha channel (transparency) into a mask, setting transparency level to 100.
5. **Bitmap Conversion:** `img.ConvertToBitmap()` converts the `wx.Image` to a `wx.Bitmap` object.
6. **Attribute Assignment:** The `exec()` statement dynamically creates an attribute of the `Images` object, using the string `name` as the attribute name and assigning the `wx.Bitmap` to it. This dynamic code generation is unsafe.
7. **_names assignment**: The _names list is assigned to a class attribute.

Data Structures

* **`_names` (list of strings):** This list contains the names of the images (which are also used as attribute names in the `Images` class).
* **`Images` (class):** This class encapsulates the loaded images. Each image (as a bitmap) is stored as an attribute of the class.

Malware Family Suggestion

While the code itself doesn't directly perform malicious actions like network communication or file system modification, its reliance on `eval()` and `exec()` makes it extremely vulnerable to code injection attacks. If an attacker were able to manipulate the contents of the `_names` list or the strings that represent the image data, they could inject arbitrary Python code that would be executed during the image loading process. This would allow for a wide range of malicious activities, potentially turning it into a downloader or dropper for a more harmful payload. Therefore, this code could be part of a:

* **Downloader/Dropper:** The Base64 encoded data could be a payload that is downloaded and executed. The use of eval() is indicative of a malicious actor attempting to hide the contents of the payload.
* **Polymorphic Malware:** The structure makes it easy to change the embedded images to change the behavior without changing the core structure.

The combination of Base64 encoding, `eval()`, and `exec()` is a strong indicator of malicious intent or, at the very least, extremely poor security practices. The code is fundamentally insecure and should not be used in any production or even testing environment without significant rewriting to remove the dangerous function calls. The `eval()` function call on the base64 strings is a severe security risk. A proper solution would involve securely loading images from trusted sources and avoiding dynamic code execution.