Analysis Report for: F9B0CE480A564EAF739A0EAFB517A3D5.exe

Overall Functionality

This C code is a batch script (indicated by `@echo off`), not a C program. It attempts to elevate its own privileges to administrator level and then install a printer. The printer installation is done using a PowerShell command encoded with Base64. The crucial part is the hidden and obfuscated PowerShell command which is likely malicious.

Function Summaries

There are no functions in the traditional C sense. This is a batch script using commands like `set`, `if`, `powershell`, and `exit`.

Control Flow

1. **Privilege Escalation Check:** The script first checks if it's already running with administrator privileges. It uses the `%1` argument to check for the presence of "am_admin". This is likely a flag internally set by the script itself after the initial elevation attempt.
2. **Privilege Elevation:** If it's not running as administrator (`if not "%1"=="am_admin"`), it uses PowerShell to restart itself with the `RunAs` verb, effectively requesting administrator privileges. This is a common technique used by malware.
3. **Printer Installation (Malicious Action):** If the script gains admin privileges, the encoded PowerShell command is executed. This part is the core malicious activity, hidden through obfuscation.
4. **Script Termination:** `exit /b` terminates the script after the printer installation (or attempted installation) is complete.

Data Structures

There are no explicit data structures used. The script uses environment variables (`ARGS1`, `ARGS2`, `ARGS`) to manipulate the script's path and handle arguments.

Malware Family Suggestion

Based on the functionality, this code strongly suggests a **backdoor or installer malware**. The hidden Base64-encoded PowerShell command is a significant red flag. It's a common tactic to hide malicious payloads. The privilege escalation attempt reinforces this suspicion; malware often needs administrative privileges to perform its malicious actions. This script likely installs a printer as a cover, masking the true intent which could be anything from remote access to data exfiltration. Further analysis of the decoded PowerShell command is needed to classify it more precisely. Without decoding the Base64 string, a definitive malware family classification is impossible. However, the techniques used strongly point towards it being a sophisticated piece of malware designed to execute malicious commands under elevated privileges.