

AppJailLauncher

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.21996.1]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>type c:\hello.txt

AppJailLauncher

C:\Windows\system32>c:\AppJaillauncher.exe /help
Usage: c:\AppJaillauncher.exe [/help] [/uninstall] {options} child-process-path
Actions:
  /help           Displays this message.
  /uninstall      By running the application jail launcher program,
                  an AppContainer profile will be created on the local
                  machine. Additionally, if a /key switch is provided,
                  the key-file-path parent directory and file will both
                  have an ACL entry added for the AppContainer SID. The
                  /uninstall switch deletes the AppContainer profile and
                  and removes ACL entries from the key-file-path parent
                  directory and file

By default, the standard action is to install the AppContainer profile and
associated access control entries if necessary and start listening as a server.
```

SSTIC Challenge

- Windows pwnable !
- Spawned by AppJailLauncher

```
root:/hnas/SSTIC# nc 192.168.1.54 4577
Menu
1. Register
2. Create maze
3. Load maze
4. Play maze
5. Remove maze
6. View scoreboard
7. Upgrade
8. Exit
```

AppJailLauncher is akin to a simple version of xinetd for Windows but with sandboxing enabled for the spawned child processes. The sandboxing is accomplished via AppContainers.

Source: https://thalium.github.io/blog/posts/sstic_infra_windows/

AppContainer Isolation

- Isolate application from logon user, devices, files, network & window
- Create a new unique “user”: AppContainer SID for an application
- Capabilities to allow access to specific objects
- Less Privileged AppContainer (LPAC): restricted AC

AppJailLauncher

- Create an AppContainer profile with no capability
- Allow read access to the flag (*cmdline parameter /key*)
- Job to limit process execution time (*cmdline parameter /timeout*)
- On connection:
 - Spawn the challenge process in the AC (with Low Integrity)
 - Redirect stdin, stdout, stderr to the client socket

AppJailLauncher Security

The SSTIC organizers forked AppJailLauncher and made some improvements:

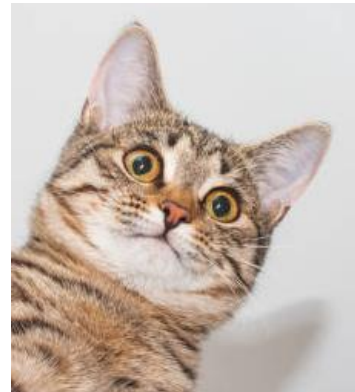
- ✓ Creates temporary new AC SID on connection (10 min timeout)
 - Previously, all the spawned challenges shared the same AC SID so after RCE, a malicious player can access other players processes (DoS)
- ✓ Limits resources by default:
 - Execution to 2 minutes
 - Memory to 100 MB of RAM
 - Maximum of 2 parallel processes
- Temporary writable folders for players (one for each player)
 - Not reviewed

Thalium Blogpost

Powershell arguments

When players executed powershell commands asking arguments from their RCE, the arguments were asked in the powershell terminal from which the AppJailLauncher were launched.

What ?



Vulnerability

```
BOOL CreateProcessW(  
    LPCWSTR      lpApplicationName,  
    LPWSTR       lpCommandLine,  
    LPSECURITY_ATTRIBUTES lpProcessAttributes,  
    LPSECURITY_ATTRIBUTES lpThreadAttributes,  
    BOOL         bInheritHandles, // TRUE  
    DWORD        dwCreationFlags, // EXTENDED_STARTUPINFO_PRESENT | CREATE_SUSPENDED  
    LPVOID       lpEnvironment,  
    LPCWSTR      lpCurrentDirectory,  
    LPSTARTUPINFOW lpStartupInfo,  
    LPPROCESS_INFORMATION lpProcessInformation  
);
```

```
LOG("Launching new process \"%s\".\n", pszCommandLine);  
W32_ASSERT(CreateProcess(  
    NULL,  
    pszCommandLine,  
    NULL,  
    NULL,  
    TRUE, // TODO: FIXME: I don't like how we're just blanket allowing all handles to be  
         // inherited.  
    dwCreationFlags, // EXTENDED_STARTUPINFO_PRESENT | CREATE_SUSPENDED  
    NULL,  
    pszCurrentDirectory,  
    (LPSTARTUPINFO) &si,  
    &pi  
), Exit);
```

```
if unsafe {  
    kernel32::CreateProcessW(cmdLine.as_ptr(),  
        0 as LPWSTR,  
        0 as LPSECURITY_ATTRIBUTES,  
        0 as LPSECURITY_ATTRIBUTES,  
        1,  
        dwCreationFlags,  
        0 as LPVOID,  
        currentDir.as_ptr(),  
        mem::transmute::<LPSTARTUPINFOEXW, LPSTARTUPINFOW>(&mut si),  
        &mut pi)  
} == 0 {
```

Basically you still have access to the console

AppJailLauncher exploit plan

- Console on Windows (condrv) uses multiple handles
 - \Input (stdin) is inherited
 - \Output (stdout, stderr) are inherited
 - \Reference is duplicated by the kernel at process creation (passed in *ProcessParameters*)
 - \Connect is created at process creation to use input/output **using \Reference handle**
 - To interact with Input/Output, you need a valid \Connect handle
- So you can control the parent console
- Exploit plan:
 - Send a Control+C event to close AppJailLauncher
 - Write a command to \Input and enjoy unsandboxed code execution!

Wait a minute!

ⓘ Note

Universal Windows Platform console apps and those with a lower integrity level than the attached console will be prohibited from both reading the output buffer and writing to the input buffer even if the security descriptors above would normally permit it. Please see the [Wrong Way Verbs](#) discussion below for more details.

The two scenarios where this can be found are:

2. Any console application intentionally launched with a lower integrity level than the existing session

Name	PID	Integrity
cmd.exe	17172	Medium
conhost.exe	9096	Medium
AppJailLauncher.exe	7452	Medium
A..Mazing.exe	11012	Low

- <https://docs.microsoft.com/en-us/windows/console/console-buffer-security-and-access-rights>

Microsoft made console open source

- Integrity level is checked to block write access to Console Input 😞

```
ConsoleProcessPolicy ConsoleProcessPolicy::s_CreateInstance(const HANDLE hProcess)
{
    bool fCanReadOutputBuffer = false;
    bool fCanWriteInputBuffer = false;
    // First check AppModel Policy:
    LOG_IF_FAILED(Microsoft::Console::Internal::ProcessPolicy::CheckAppModelPolicy(hToken.get(), fIsWrongWayBlocked));

    // If we're not restricted by AppModel Policy, also check for Integrity Level below our own.
    if (!fIsWrongWayBlocked)
    {
        LOG_IF_FAILED(Microsoft::Console::Internal::ProcessPolicy::CheckIntegrityLevelPolicy(hToken.get(), fIsWrongWayBlocked));
    }

    // If we're not blocking wrong way verbs, adjust the read/write policies to permit read out and write in.
    if (!fIsWrongWayBlocked)
    {
        fCanReadOutputBuffer = true;
        fCanWriteInputBuffer = true;
    }
    // ...
}
```

- <https://github.com/microsoft/terminal/blob/main/src/server/ProcessPolicy.cpp#L27>

Let's check the binary (21H1)

Where is the integrity level check ?

```
ConsoleProcessPolicy *__fastcall ConsoleProcessPolicy::s_CreateInstance(ConsoleProcessPolicy *a1, void *hProcess)
{
    bool fIsAuthorized; // di
    char *v5; // rcx
    char fIsWrongWayBlocked; // [rsp+30h] [rbp+8h] BYREF
    void *TokenHandle; // [rsp+40h] [rbp+18h] BYREF

    fIsAuthorized = 0;
    TokenHandle = 0i64;
    __();
    if ( OpenProcessToken(hProcess, 0x20008u, &TokenHandle) )
    {
        fIsWrongWayBlocked = 1;
        if ( ConsoleProcessPolicy::s_CheckAppModelPolicy(TokenHandle, &fIsWrongWayBlocked) < 0 )
            __();
        if ( !fIsWrongWayBlocked )
        {
            if ( ConsoleProcessPolicy::s_CheckAppModelPolicy(TokenHandle, &fIsWrongWayBlocked) < 0 )
                __();
            fIsAuthorized = fIsWrongWayBlocked == 0;
        }
    }
    else
    {
        __();
    }
    v5 = TokenHandle;
    a1->_fCanReadOutputBuffer = fIsAuthorized;
    a1->_fCanWriteInputBuffer = fIsAuthorized;
    ...
}
```

*_: Removed some names for simplicity

Exploitability

Windows Version	Can access console	Can write input buffer
Before 1803	No (AC can't create connection)	No
1803	Yes	No (Check Integrity)
1809	Yes	No (Check Integrity)
1903	Yes	No (Check Integrity)
1909	Yes	No (Check Integrity)
2004	Yes	Yes
20H2	Yes	Yes
21H1 (current version)	Yes	Yes
Windows 11	Yes	No (Fixed again)

Writing to the console is fixed and disabled but escaping may be possible using other commands (AddConsoleAlias)



Demo !

Demo !

```
mastho@DESKTOP-PLI697L:/mnt/c/Windows/system32$ nc 127.0.0.1 1337
Flag: FLAG_THIS_IS_THE_FLAG
Trying to overwrite flag.txt
Cannot overwrite flag.txt
Flag: FLAG_THIS_IS_THE_FLAG
Sending command injection to handle 84 mode 503
Sending command injection to handle 88 mode 3
Sending command injection to handle 92 mode 3
Checking if payload was executed
Flag: REPLACED
Press enter to exit
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mastho\Downloads\AppJailLauncher-master\Clean>echo FLAG_THIS_IS_THE_FLAG > flag.txt

C:\Users\mastho\Downloads\AppJailLauncher-master\Clean>AppJailLauncher.exe /key:flag.txt /port:1337 /timeout:12000000 exploit.exe
Listening for incoming connections on port 1337...
Client connection from 127.0.0.1 accepted.
Ctrl-C event detected. Exiting...
Goodbye.

C:\Users\mastho\Downloads\AppJailLauncher-master\Clean>echo REPLACED > flag.txt

C:\Users\mastho\Downloads\AppJailLauncher-master\Clean>
```

Left as an exercise for the reader:

“ During the competition, this command was launched with Powershell. ”

AppJailLauncher Fix

- Process Creation Flags
 - **CREATE_NEW_CONSOLE**
 - The *new process* has a new console, instead of inheriting its parent's console (the default).

```
2 AppJailLauncher/utils.cpp
@@ -383,7 +383,7 @@ HRESULT CreateClientSocketWorker(
383
384     LPTSTR pszCommandLine = NULL;
385     PSID pSid = NULL;
386 -     DWORD dwCreationFlags = CREATE_SUSPENDED;
386 +     DWORD dwCreationFlags = CREATE_SUSPENDED | CREATE_NEW_CONSOLE;
387     DWORD dwCapabilitiesCount = 0;
388     DWORD dwAttributeListSize = 0;
```

Without the parent console \Reference, inherited Input/Output are unusable.

<https://t.me/learningshorts> [Commit: 4764645](#) and [dd37034](#)

Microsoft Fix

- October 2021: [CVE-2021-41346](#)

```
ConsoleProcessPolicy * __fastcall ConsoleProcessPolicy::s_CreateInstance(ConsoleProcessPolicy *ret, void *hProcess)
{
    bool fIsAuthorized; // di
    char fIsWrongWayBlocked; // [rsp+30h] [rbp+8h] BYREF
    void *TokenHandle; // [rsp+40h] [rbp+18h] MAPDST BYREF

    fIsAuthorized = 0;
    TokenHandle = 0i64;
    __();
    if ( OpenProcessToken(hProcess, 0x20008u, &TokenHandle) )
    {
        fIsWrongWayBlocked = 1;
        if ( (int)ConsoleProcessPolicy::s_CheckAppModelPolicy(TokenHandle, &fIsWrongWayBlocked) < 0 )
            __();
        if ( !fIsWrongWayBlocked )
        {
            if ( (int)ConsoleProcessPolicy::s_CheckIntegrityLevelPolicy(TokenHandle, &fIsWrongWayBlocked) < 0 )
                __();
            fIsAuthorized = fIsWrongWayBlocked == 0;
        }
    }
    else
    {
        __();
    }
    ret->_fCanReadOutputBuffer = fIsAuthorized;
    ret->_fCanWriteInputBuffer = fIsAuthorized;
}
```

Links

- AppJailLauncher console escape exploit:

<https://gist.github.com/masthoon/3b3b60dcb7f8687dc7336bcbe3236700>

- POC [CVE-2021-41346](#)

<https://gist.github.com/masthoon/85fab432527329f17a040b311fc1f2a2>