THALIUM

Fuzzing Microsoft's RDP Client using Virtual Channels

Valentino RICOTTA

1er juin 2022

SSTIC SYMPOSIUM
SUR LA SÉCURITÉ DES TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Introduction

`whoami`

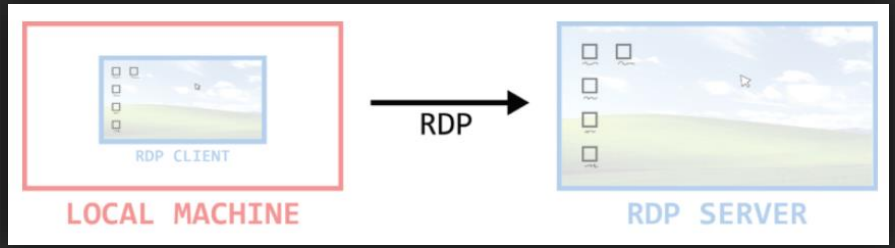
- Étudiant @ CentraleSupélec
- Passionné de reverse, sécu offensive, crypto…
- Joueur de CTF insatiable
  - 1ère place au France Cyber Security Challenge 2022

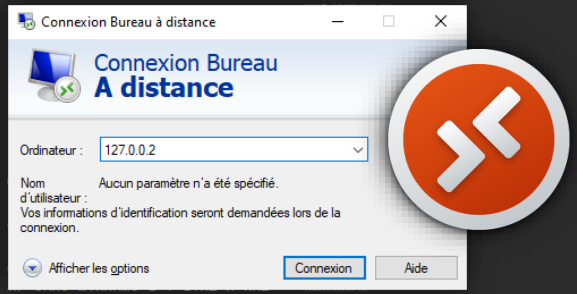
- 2021 : Stage de césure @ Thalium
- 2022 : Stage de fin d'études @ Thalium

Contexte et motivation

Remote Desktop Protocol (RDP)

- Protocole propriétaire de Microsoft
- Connexion à un ordinateur à distance via interface graphique





Contexte et motivation

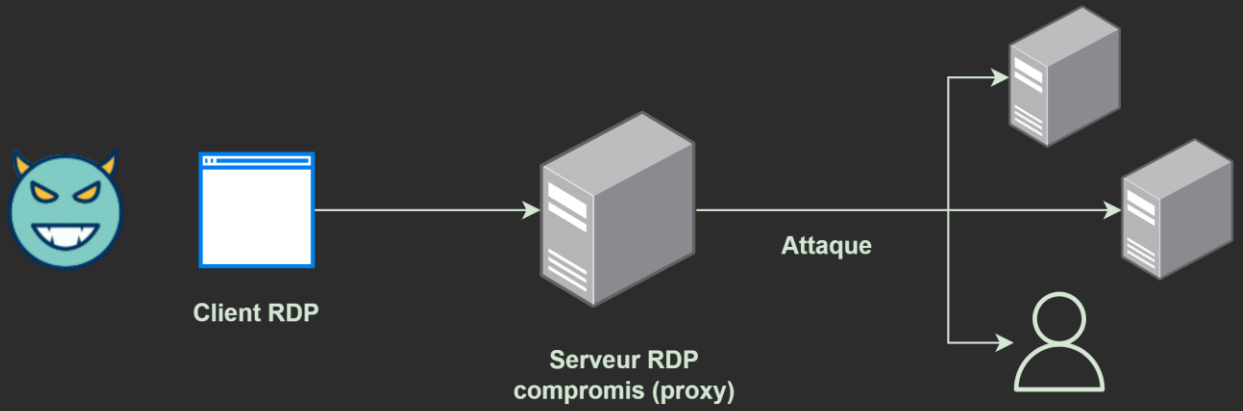
Travaux existants

- 2019 : *Fuzzing and Exploiting Virtual Channels in Microsoft Remote Desktop Protocol for Fun and Profit* (Blackhat Europe 2019)
  - RCE dans le client RDP en fuzzant les *Virtual Channels* avec WinAFL
- 2021 : *Fuzzing RDP: Holding the Stick at Both Ends* (CyberArk)
  - Sujet similaire — approche, fuzzing et résultats différents

Contexte et motivation

Pourquoi s’intéresser au *client* RDP ?

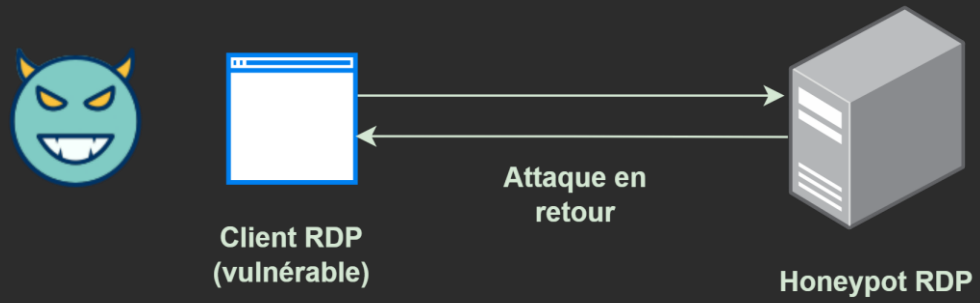
- Scénario « honeypot » proposé à la Blackhat



Contexte et motivation

Pourquoi s’intéresser au *client* RDP ?

- Scénario « honeypot » proposé à la Blackhat



Contexte et motivation

Pourquoi s’intéresser au *client* RDP ?

- Scénario « honeypot » proposé à la Blackhat

- Surface d’attaque aussi riche que côté serveur
- Moins de recherche

Contexte et motivation

Pourquoi s’intéresser au *client* RDP ?

- Scénario « honeypot » proposé à la Blackhat

- Surface d’attaque aussi riche que côté serveur
- Moins de recherche

- Guest-to-host Hyper-V escapes
  - *Depuis quelques mois MSRC n’accepte plus les vulnérabilités RDP dans son Hyper-V Bounty Program…*

Sommaire

1. Étude du protocole RDP
2. Architecture d’un fuzzer pour clients RDP
3. Stratégie de fuzzing
4. Vulnérabilités identifiées
5. Conclusion

Étude du protocole RDP

Étude du protocole RDP

Client RDP

- Deux binaires d'intérêt : mstsc.exe et mstscax.dll

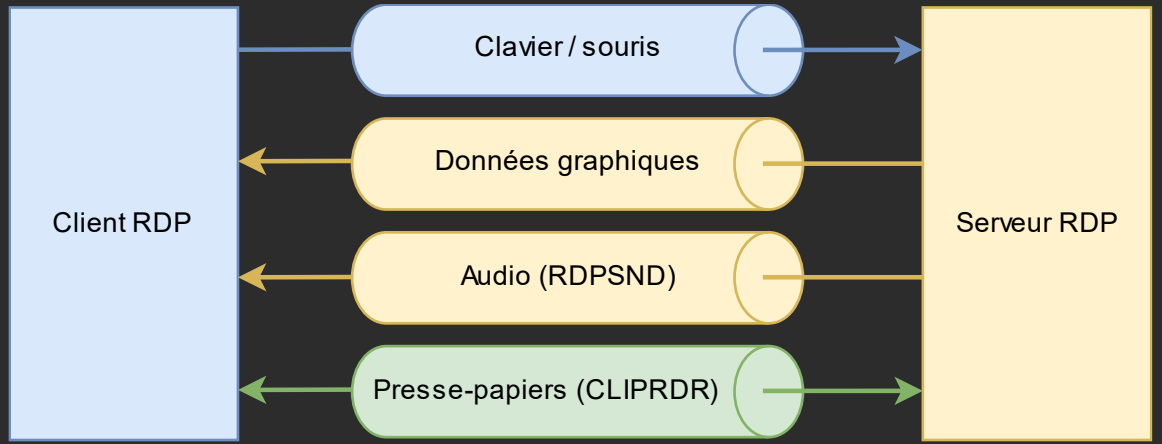
- Fonctionnalités basiques :
  - Recevoir l'image du bureau
  - Envoyer des inputs clavier et souris

- *Virtual Channels*
  - Extensions logicielles
  - Améliorations fonctionnelles, support hardware…
  - Un bon nombre qui sont présentes par défaut

Étude du protocole RDP

Virtual Channels

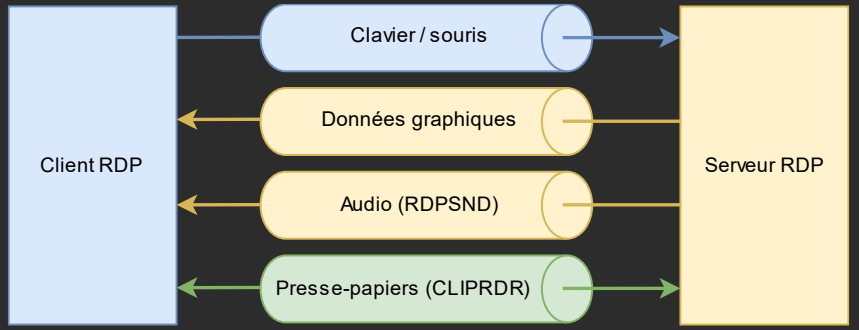
- Couche d'abstraction qui permet de transporter des données
- Chaque *virtual channel* possède sa propre logique, spécification, protocole



Étude du protocole RDP

Virtual Channels

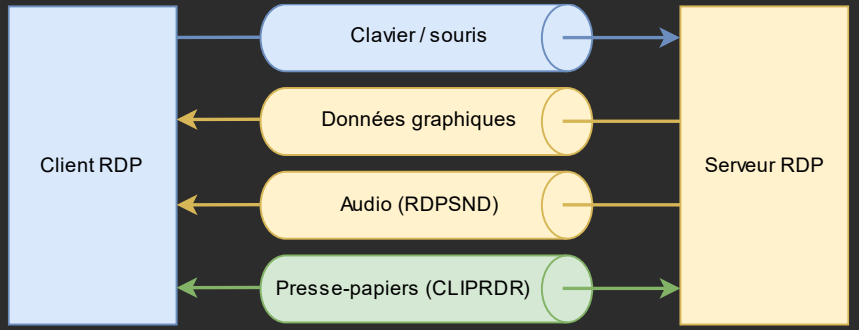
- Des channels statiques :
  - `CLIPRDR`, `RDPSND`, `RDPDR`
  - `DRDYNVC` (support des channels dynamiques)



Étude du protocole RDP

Virtual Channels

- Des channels statiques :
  - `CLIPRDR`, `RDPSND`, `RDPDR`
  - `DRDYNVC` (support des channels dynamiques)



- Des channels dynamiques :
  - Utilisés par les développeurs pour des extensions (et par certains attaquants…)
  - Audio Input, Video Redirection, PnP, Display Control, Telemetry…

Étude du protocole RDP

Virtual Channels

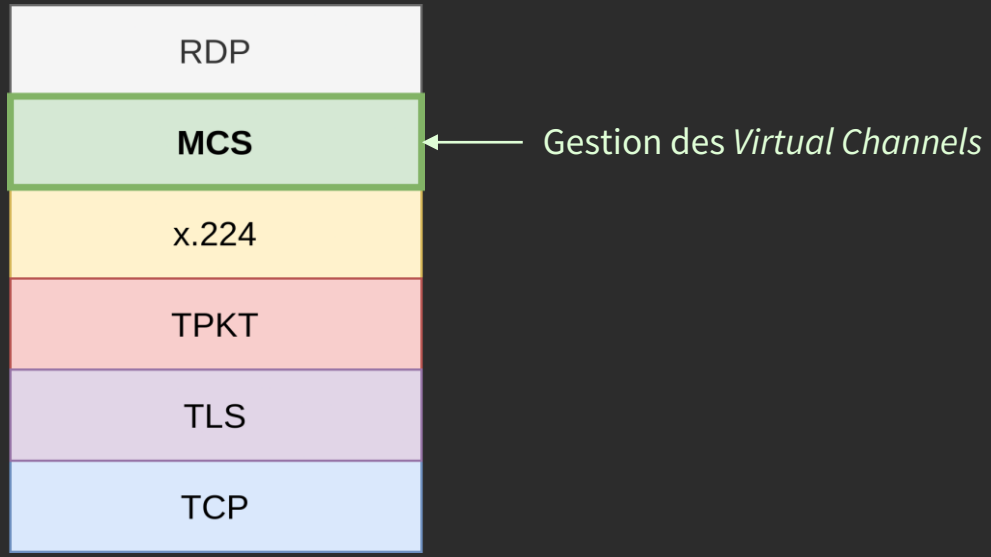
- Des channels statiques :
  - `CLIPRDR`, `RDPSND`, `RDPDR`
  - `DRDYNVC` (support des channels dynamiques)

} Ouverts par défaut

- Des channels dynamiques :
  - Utilisés par les développeurs pour des extensions (et par certains attaquants…)
  - Audio Input, Video Redirection, PnP, Display Control, Telemetry…

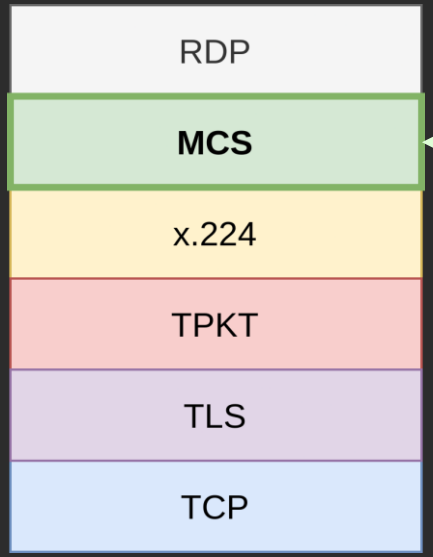
Étude du protocole RDP

Pile réseau



Étude du protocole RDP

Pile réseau



Gestion des *Virtual Channels*

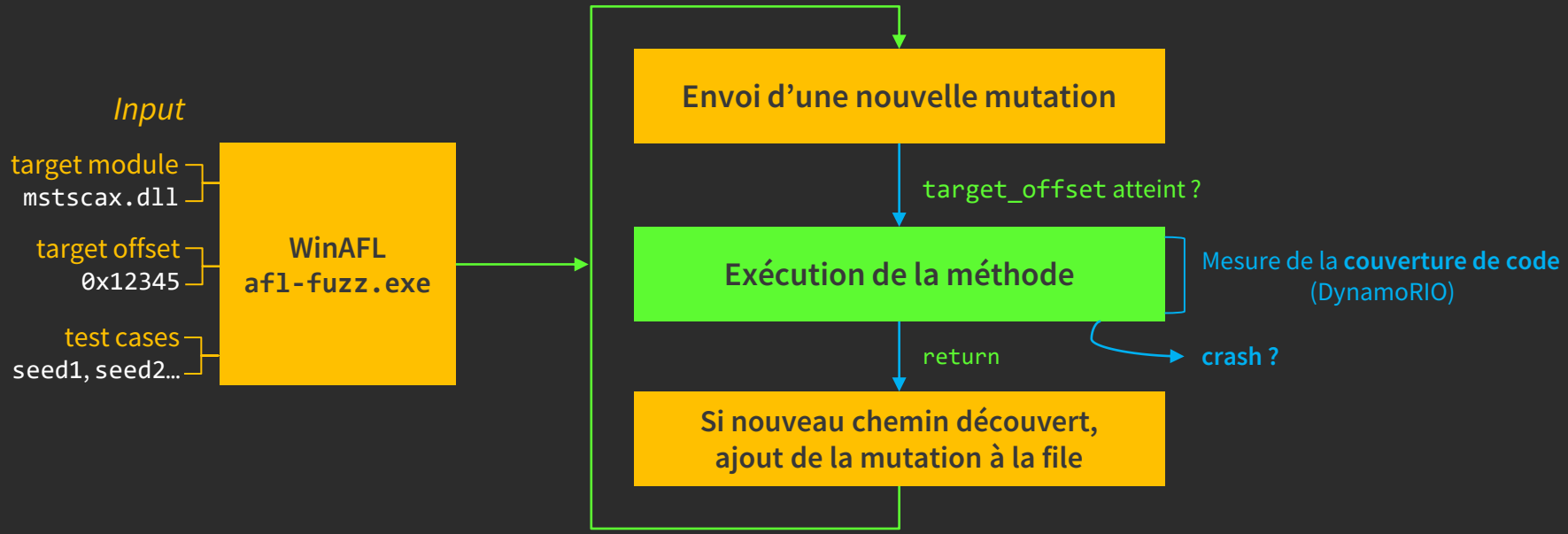
WTSAPI32

- WTSVirtualChannelClose function
- WTSVirtualChannelOpen function
- WTSVirtualChannelOpenEx function
- WTSVirtualChannelPurgeInput function
- WTSVirtualChannelPurgeOutput function
- WTSVirtualChannelQuery function
- WTSVirtualChannelRead function
- WTSVirtualChannelWrite function

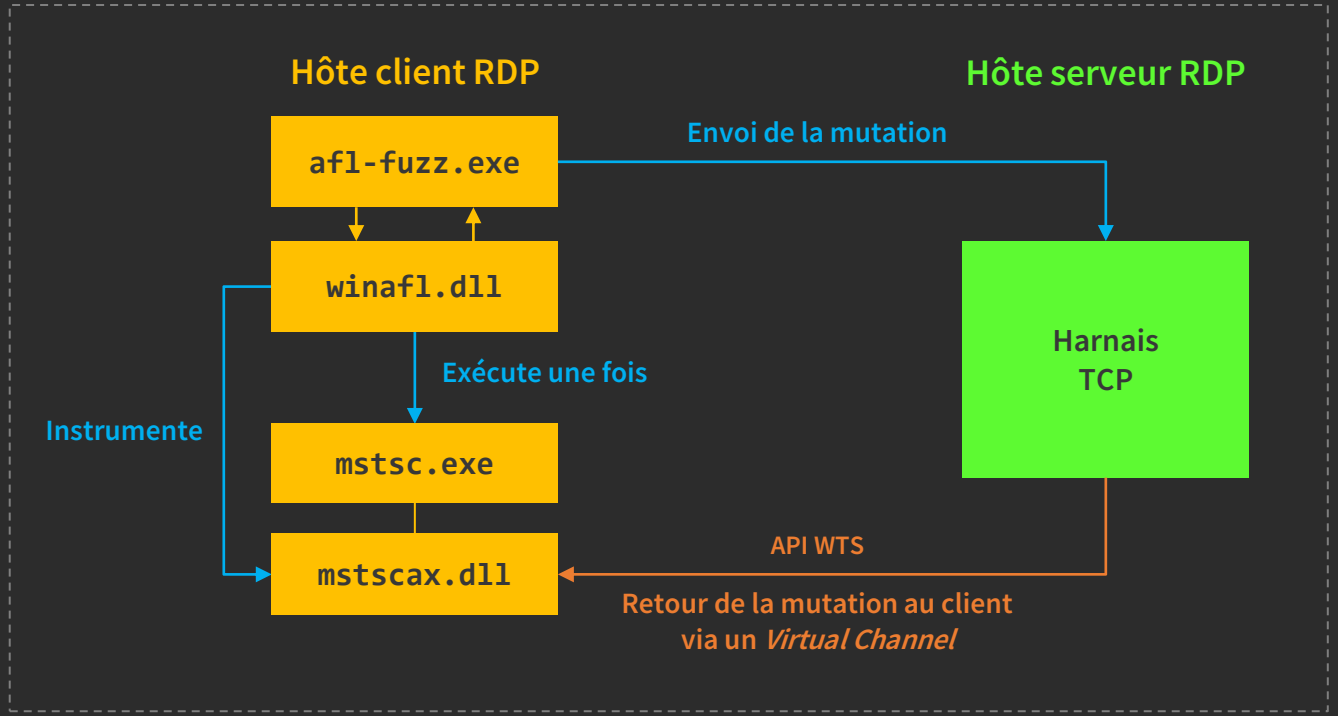
Architecture d’un fuzzer pour clients RDP

Architecture d’un fuzzer basé sur WinAFL

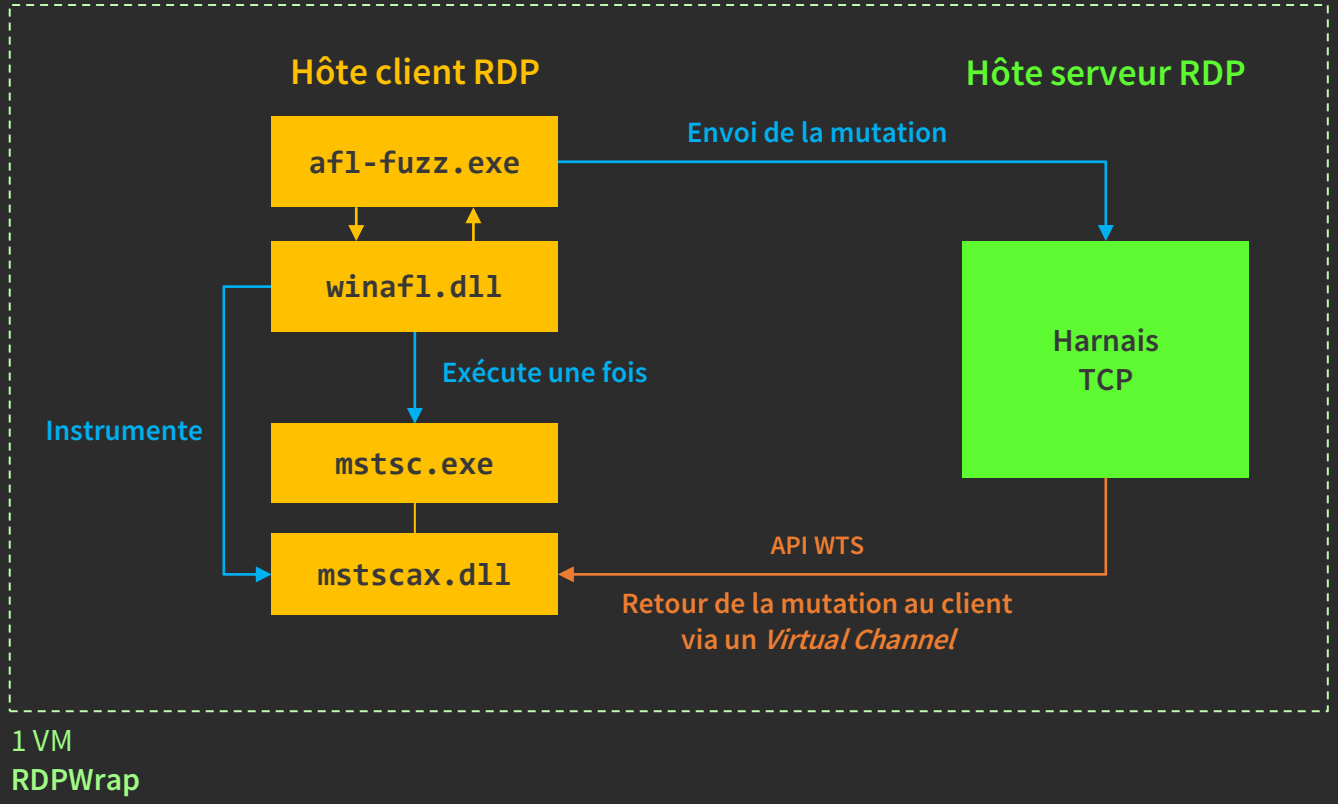
Fonctionnement simplifié de WinAFL



Architecture d'un fuzzer basé sur WinAFL



Architecture d'un fuzzer basé sur WinAFL



Stratégie de fuzzing

Stratégie de fuzzing

Buts

- Construire une approche permettant de fuzzer rapidement une liste de *channels*
- Fuzzer ces *channels* avec une couverture décente

Stratégie de fuzzing

Comment s’attaquer à un channel ?

- Premier channel étudié : RDPSND
- Lecture de la spécification officielle de Microsoft
  - Description du fonctionnement

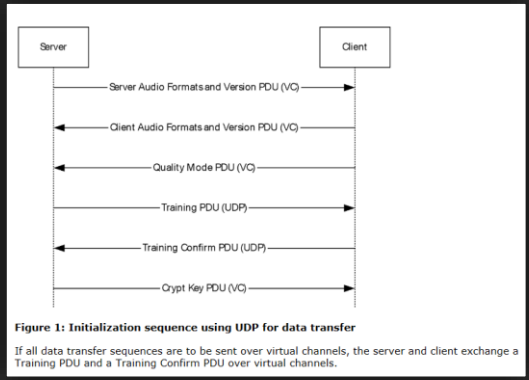


Figure 1: Initialization sequence using UDP for data transfer

If all data transfer sequences are to be sent over virtual channels, the server and client exchange a Training PDU and a Training Confirm PDU over virtual channels.

Stratégie de fuzzing

Comment s'attaquer à un channel ?

- Premier channel étudié : RDPSND
- Lecture de la spécification officielle de Microsoft
  - Description du fonctionnement
  - Types de messages, structures de données

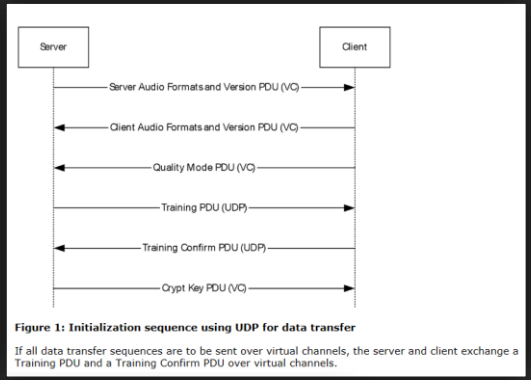


Figure 1: Initialization sequence using UDP for data transfer

If all data transfer sequences are to be sent over virtual channels, the server and client exchange a Training PDU and a Training Confirm PDU over virtual channels.

2.2 Message Syntax

The following sections contain Remote Desktop Protocol: Audio Output Virtual Channel Extension message syntax.

2.2.1 RDPSND PDU Header (SNDPROLOG)

The RDPSND PDU header is present in many audio **PDUs**. It is used to identify the PDU type, specify the length of the PDU, and convey message flags.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
msgType								bPad								BodySize															

msgType (1 byte): An 8-bit unsigned integer that specifies the type of audio PDU that follows the **BodySize** field.

Value	Meaning
SNDC_CLOSE 0x01	Close PDU
SNDC_WAVE 0x02	WaveInfo PDU
SNDC_SETVOLUME 0x03	Volume PDU
SNDC_SETPITCH	Pitch PDU

Stratégie de fuzzing

Comment s'attaquer à un channel ?

- Premier channel étudié : RDPSND
- Lecture de la spécification officielle de Microsoft
  - Description du fonctionnement
  - Types de messages, structures de données
  - Exemples de PDU (*Protocol Data Unit*) → seeds !

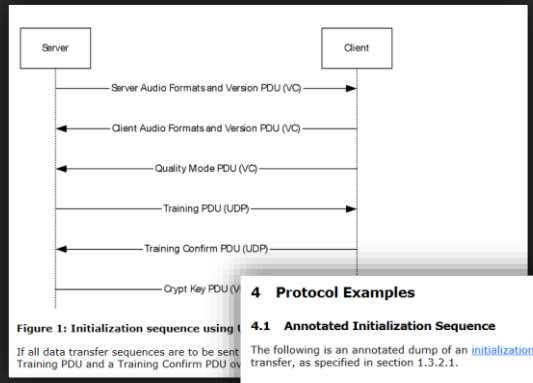


Figure 1: Initialization sequence using t

If all data transfer sequences are to be sent
Training PDU and a Training Confirm PDU ov

4 Protocol Examples

4.1 Annotated Initialization Sequence

The following is an annotated dump of an initialization sequence using virtual channels for data transfer, as specified in section 1.3.2.1.

4.1.1 Server Audio Formats and Version PDU

The following is an annotated dump of a Server Audio Formats and Version PDU.

```
00000000 07 2b 90 00 08 fb 8b 00 e0 f1 09 00 70 27 1f 77  .+..........p'.w
00000010 00 00 05 00 ff 05 00 00 01 00 02 00 22 56 00 00  ............"V..
00000020 88 58 01 00 04 00 10 00 00 00 06 00 02 00 22 56  .X............"V
00000030 00 00 44 ac 00 00 02 00 08 00 00 00 07 00 02 00  ..D.............
00000040 22 56 00 00 44 ac 00 00 02 00 08 00 00 00 02 00  "V..D...........
00000050 02 00 22 56 00 00 27 57 00 00 00 00 04 04 00 20  ..."V..'W.......
00000060 f4 03 07 00 00 01 00 00 00 02 00 ff 00 00 00 00  ................
00000070 c0 00 40 00 f0 00 00 00 cc 01 30 ff 88 01 18 ff  ..@.......0.....
00000080 11 00 02 00 22 56 00 00 b9 56 00 00 00 04 04 00  ....."V...V......
00000090 02 00 f9 03

07 -> SNDPROLOG::bType = SNDC_FORMATS (7)
2b -> SNDPROLOG::bPad = 0x2b
90 00 -> SNDPROLOG::wBodySize = 0x90 = 144 bytes

08 fb 8b 00 -> SERVER_AUDIO_VERSION_AND_FORMATS::dwFlags = 0x008bfb08
e0 f1 09 00 -> SERVER_AUDIO_VERSION_AND_FORMATS::dwVolume = 0x0009f1e0
70 27 1f 77 -> SERVER_AUDIO_VERSION_AND_FORMATS::dwPitch = 0x771f2770
00 00 -> SERVER_AUDIO_VERSION_AND_FORMATS::wDGramPort = 0
05 00 -> SERVER_AUDIO_VERSION_AND_FORMATS::wNumberOfFormats = 5
ff -> SERVER_AUDIO_VERSION_AND_FORMATS::cLastBlockConfirmed = 0xff = 255
05 00 -> SERVER_AUDIO_VERSION_AND_FORMATS::wVersion = 5
00 -> SERVER_AUDIO_VERSION_AND_FORMATS::bPad = 0
01 00 02 00 22 56 00 00 88 58 01 00 04 00 10 00 00 00 -> AUDIO_FORMAT
01 00 -> AUDIO_FORMAT::wFormatTag = WAVE_FORMAT_PCM (1)
   02 00 -> AUDIO_FORMAT::nChannels = 2
   22 56 00 00 -> AUDIO_FORMAT::nSamplesPerSec = 0x5622 = 22050
   88 58 01 00 -> AUDIO_FORMAT::nAvgBytesPerSec = 0x15888 = 88200
   04 00 -> AUDIO_FORMAT::nBlockAlign = 0x0004 = 4
   10 00 -> AUDIO_FORMAT::wBitsPerSample = 0x10 = 16
   00 00 -> AUDIO_FORMAT::cbSize = 0
```

2.2 Message Syntax

The following sections contain Remote Desktop Protocol: Audio Output Virtual Channel message syntax.

2.2.1 RDPSND PDU Header (SNDPROLOG)

The RDPSND PDU header is present in many audio **PDUs**. It is used to identify the PD the length of the PDU, and convey message flags.

msgType (1 byte): An 8-bit unsigned integer that specifies the type of audio PDU th
BodySize field.

Value	Meaning
SNDC_CLOSE 0x01	Close PDU
SNDC_WAVE 0x02	WaveInfo PDU
SNDC_SETVOLUME 0x03	Volume PDU
SNDC_SETPITCH	Pitch PDU

Stratégie de fuzzing

Comment s'attaquer à un channel ?

- Premier channel étudié : RDPSND
- Lecture de la spécification officielle de Microsoft
  - Description du fonctionnement
  - Types de messages, structures de données
  - Exemples de PDU (*Protocol Data Unit*) → seeds !
- Localiser le handler des PDUs pour chaque channel

```
int64 __fastcall CRdpAudioController::DataArrived(
  __int64 this,
  unsigned __int8 *PDU,
  _DWORD *a3,
  unsigned int a4
)
```

```
switch (PDU->Header.msgType) {
  case 0x01: ...
  case 0x02: ...
  case 0x03: ...
}
```

Stratégie de fuzzing

Différentes stratégies de fuzzing

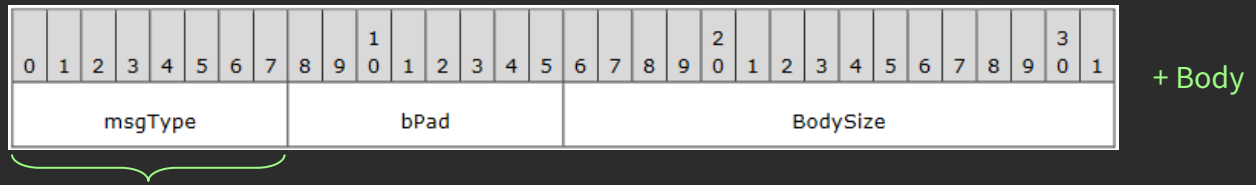
- Problématique du fuzzing de machines à états
- Deux stratégies proposées
  - Chacune a ses avantages et inconvénients
  - Aucune ne règle *vraiment* le problème

Stratégie de fuzzing

1. Mixed message type fuzzing

- Stratégie « naïve »
  - Les *seeds* obtenues de la spécification sont utilisées de façon brute
  - Pas de traitement côté harnais

Header « SNDPROLOG »



+ Body

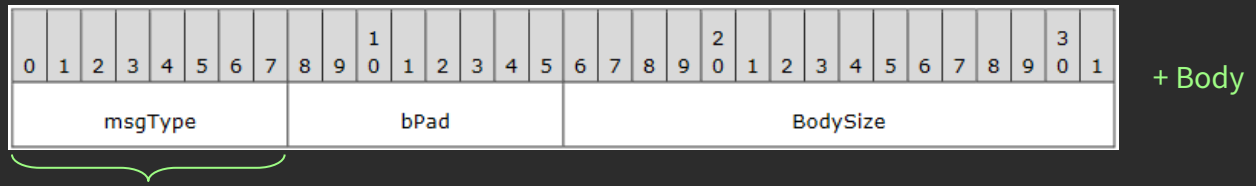
0x01 SNDC_CLOSE
0x02 SNDC_WAVE
0x03 SNDC_SETVOLUME
…

Stratégie de fuzzing

1. Mixed message type fuzzing

- Stratégie « naïve »
  - Les *seeds* obtenues de la spécification sont utilisées de façon brute
  - Pas de traitement côté harnais

Header « SNDPROLOG »



+ Body

0x01 SNDC_CLOSE
0x02 SNDC_WAVE
0x03 SNDC_SETVOLUME
…

Les `msgType` sont mutés.

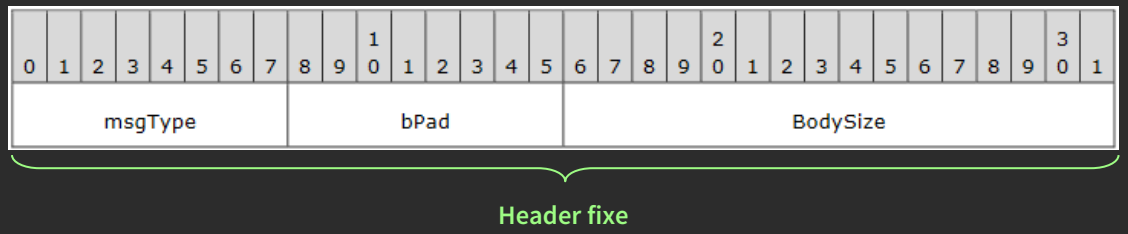
- **Mélange** des types de message lors du fuzzing
- Apparition de bugs *stateful*
- Analyse de crash plus difficile

Stratégie de fuzzing

2. Fixed message type fuzzing

- Les différents types de messages sont fuzzés séparément.

Header « SNDPROLOG »



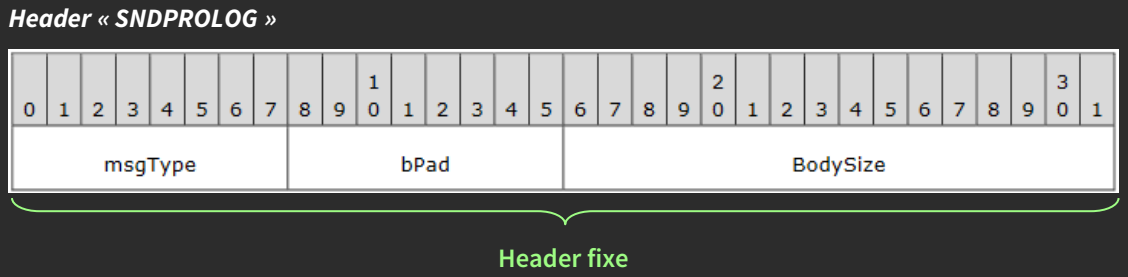
+ Body

Le fuzzer n'agit qu'ici

Stratégie de fuzzing

2. Fixed message type fuzzing

- Les différents types de messages sont fuzzés séparément.



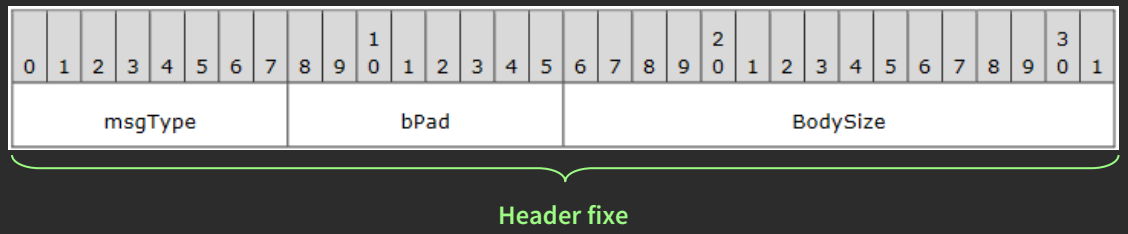
- Beaucoup moins de bugs *stateful*
- Un peu moins de bugs
- Plus faciles à analyser

Stratégie de fuzzing

2. Fixed message type fuzzing

- Les différents types de messages sont fuzzés séparément.

Header « SNDPROLOG »



+ Body

Le fuzzer n'agit qu'ici

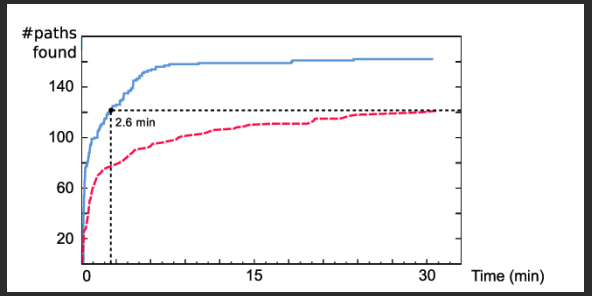
- Beaucoup moins de bugs *stateful*
- Un peu moins de bugs
- Plus faciles à analyser

Exemple : **dispatchs asynchrones** dans `CLIPRDR`

Stratégie de fuzzing

Qualité du fuzzing

- On a fuzzé un channel pendant 72 heures…
- Quand est-ce qu'on s'arrête ?
- Qualité de la couverture ?
  - Modifier WinAFL pour logger les nouveaux chemins
  - IDA + Lighthouse pour analyser la couverture de *basic blocks*



Cov %	Func Name	Address	Blocks Hit	Instr. Hit	Func Size	CC
11.13	CRdpAudioController::OnWaveData(void *,void *,_XBool32)	0x6B760	111 / 256	141 / 1267	5789	173
15.76	CRdpAudioController::DataArrived(void *,void *,_XBool32)	0x6ADA0	60 / 108	78 / 495	2070	75
12.97	CAudioConverter::OpenConverter(tWAVEFORMATEX *,tWAVEFORMATEX *,HACMDRIVERID__ *)	0x37F690	52 / 104	55 / 424	1794	77
13.53	CRdpAudioController::ChooseSoundFormat(ulong,SNDFORMATITEM *,SNDFORMATITEM * *,ulong *,ulong *)	0x15CB00	50 / 96	59 / 436	1859	64
12.13	CAudioConverter::Convert(uchar *,ulong,uchar *,ulong *)	0x37E758	43 / 92	53 / 437	1805	63
8.26	CAudioMaster::OpenConverter(tWAVEFORMATEX *,tWAVEFORMATEX *,HACMDRIVERID__ *)	0x37D218	35 / 139	46 / 557	2377	100
10.34	CRdpAudioController::UpdateDataBufferedInDevice(ulong)	0x2226C	27 / 56	30 / 290	1304	39
9.38	CChan::IntVirtualChannelWrite(ulong,void *,ulong,void *)	0xA47FC	26 / 70	32 / 341	1420	47
11.27	FindSuggestedConverter(HACMDRIVERID__ *,tWAVEFORMATEX *,tWAVEFORMATEX *,tWAVEFORMATEX *,int *)	0x37EE6C	23 / 57	31 / 275	1169	41
10.75	CRdpAudioController::GetRemotePresentationTime(__int64 *)	0x6CE10	20 / 50	23 / 214	988	36
12.50	CRdpAudioController::UpdateAndGetDataBufferedInDeviceInfo(uchar *,ushort *,ulong *)	0x6E294	20 / 47	24 / 192	826	32
7.00	CRDPAudioVideoSyncHandler::GetAggregatedLagForAStream(ulong,__int64 *)	0x6D6E0	19 / 75	21 / 300	1300	49
12.92	CRdpAudioController::DetectGlitch(void)	0x6DC08	19 / 34	23 / 178	779	23
5.12	CDynVCPlugin::SendChannelData(CWriteBuffer *)	0x7D298	18 / 96	24 / 469	1944	61
6.03	MapHRToXResult(long)	0xF3D0	17 / 140	17 / 282	963	13
9.62	CRdpWinAudioWaveoutPlayback::Render(uchar,ushort,signed char *,uint)	0x2C360	17 / 42	20 / 208	951	31
9.80	CRdpAudioController::OnNewFormat(ulong)	0x15E970	16 / 35	20 / 204	857	25
18.27	CPCMDriverCache::GetDrivers(tWAVEFORMATEX *,HACMDRIVERID__ * *,HACMDRIVERID__ * *)	0x37F304	16 / 20	19 / 104	400	15
10.21	CDynVCChannel::Write(ulong,uchar *,IUnknown *)	0x7CE20	15 / 44	24 / 235	1023	28
11.49	CRDPAudioVideoSyncHandler::GetAggregatedLag(__int64 *)	0x6D44C	14 / 30	17 / 148	653	20

Proportion de basic blocks couverts

Vulnérabilités identifiées

Vulnérabilités identifiées

Résultats des campagnes de fuzzing

Channel	**Description**	**Bugs**
`RDPSND`	Redirection audio serveur → client	1
`CLIPRDR`	Synchronisation presse-papiers	1
`RDPDR`	Redirection *filesystem*, imprimantes, cartes à puce…	3

RDPDR

RDPDR

- Accès au *filesystem* du client depuis le serveur
- Extensions (*sub-protocols*) :
  - *Smart Cards*
  - *Printers*
  - *Ports (Serial/Parallel)*

RDPDR

- Accès au *filesystem* du client depuis le serveur
- Extensions (*sub-protocols*) :
  - *Smart Cards*
  - *Printers*
  - *Ports (Serial/Parallel)*

- Résultats
  - Un bug « *Arbitrary Malloc DoS* »
  - Remote Heap Leak (CVE-2021-38665) dans l'extension *Printer*
  - Remote Heap Buffer Overflow (CVE-2021-38666) dans l'extension *Smart Card*

Remote Heap Leak (CVE-2021-38665)

Découverte du bug

- En train de fuzzer RDPDR
- Des millions d’exécutions…

Remote Heap Leak (CVE-2021-38665)

Découverte du bug

- En train de fuzzer RDPDR
- Des millions d’exécutions…

- Client RDP cassé
  - Remplit toute la RAM à **chaque démarrage**

Remote Heap Leak (CVE-2021-38665)

Découverte du bug

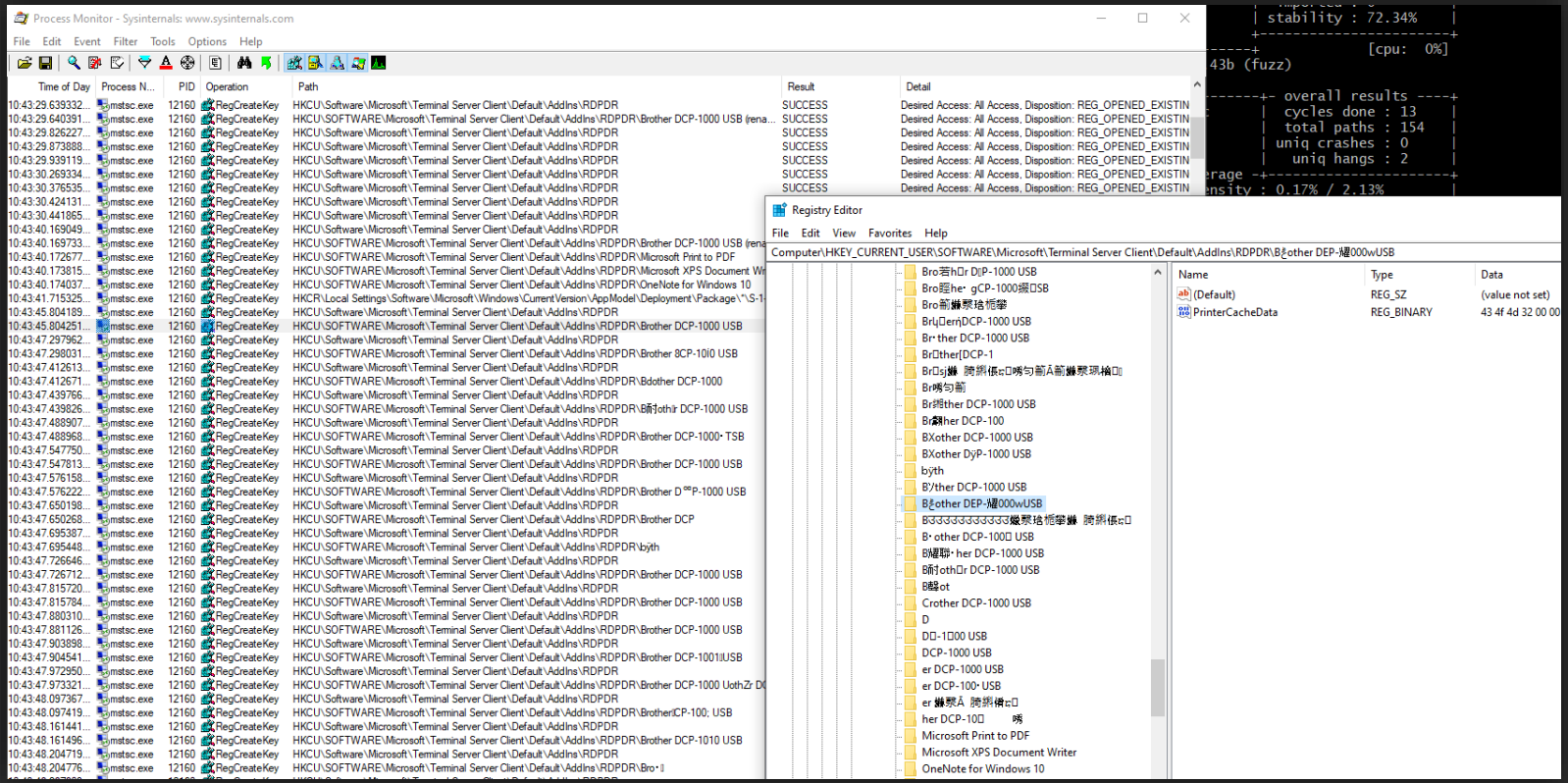
- En train de fuzzer RDPDR
- Des millions d'exécutions…

- Client RDP cassé
  - Remplit toute la RAM à **chaque démarrage**

- Les effets secondaires du fuzzing sur le système peuvent révéler des bugs !

Remote Heap Leak (CVE-2021-38665)

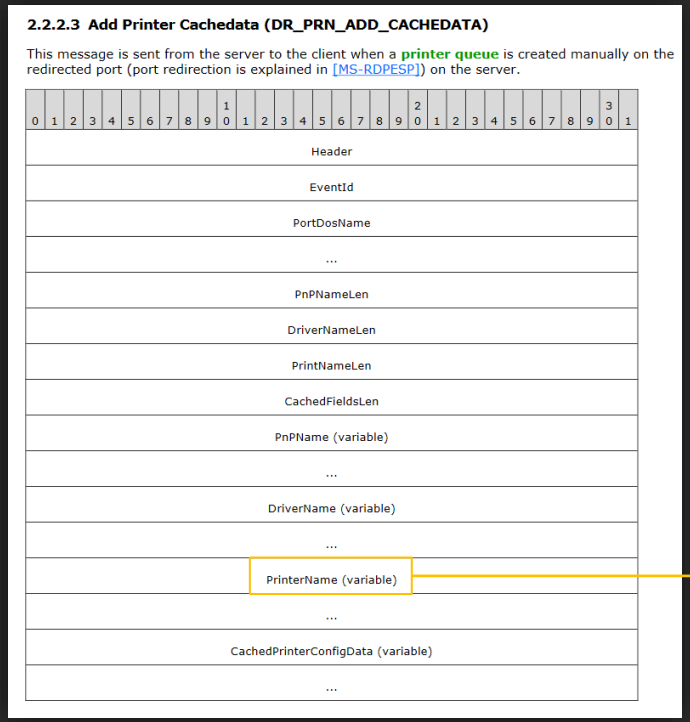
Découverte du bug



Remote Heap Leak (CVE-2021-38665)

Le coupable

- « *Add Printer Cachedata* » (DR_PRN_ADD_CACHEDATA)



2.2.2.3 Add Printer Cachedata (DR_PRN_ADD_CACHEDATA)

This message is sent from the server to the client when a **printer queue** is created manually on the redirected port (port redirection is explained in [MS-RDPESP]) on the server.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Header																															
EventId																															
PortDosName																															
...																															
PnPNameLen																															
DriverNameLen																															
PrintNameLen																															
CachedFieldsLen																															
PnPName (variable)																															
...																															
DriverName (variable)																															
...																															
PrinterName (variable)																															
...																															
CachedPrinterConfigData (variable)																															
...																															

```
RegCreateKeyExW(
        hKey,
        PrinterName,
        0,
        0,
        0,
        0xF003F,
        0,
        &phkResult,
        &dwDisposition
)
```

- **Nom de clé de registre arbitraire**
- **Pas de check de longueur**
- **Pas de double null byte (00 00)**

Remote Heap Leak (CVE-2021-38665)

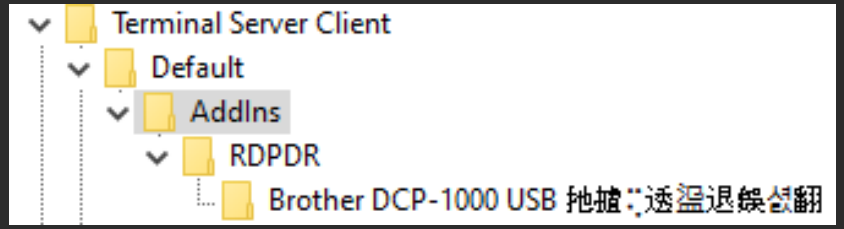
Heap Leak

```
char leak_heap[] = {
  // DR_PRN_ADD_CACHEDATA
  0x52, 0x50, 0x43, 0x50,                         // Header
  0x01, 0x00, 0x00, 0x00,                         // EventId
  0x43, 0x4f, 0x4d, 0x32, 0x00, 0x00, 0x3a, 0x00, // PortDosName
  0x00, 0x00, 0x00, 0x00,                         // PnpNameLen
  0x2a, 0x00, 0x00, 0x00,                         // DriverNameLen
  0x2a, 0x00, 0x00, 0x00,                         // PrintNameLen
  0x00, 0x00, 0x00, 0x00,                         // CachedFieldsLen
  // DriverName
  0x42, 0x00, 0x72, 0x00, 0x6f, 0x00, 0x74, 0x00, 0x68, 0x00, 0x65, 0x00, 0x72, 0x00,
  0x20, 0x00, 0x44, 0x00, 0x43, 0x00, 0x50, 0x00, 0x2d, 0x00, 0x31, 0x00, 0x30, 0x00,
  0x30, 0x00, 0x30, 0x00, 0x20, 0x00, 0x55, 0x00, 0x53, 0x00, 0x42, 0x00, 0x00, 0x00,
  // PrinterName
  0x42, 0x00, 0x72, 0x00, 0x6f, 0x00, 0x74, 0x00, 0x68, 0x00, 0x65, 0x00, 0x72, 0x00,
  0x20, 0x00, 0x44, 0x00, 0x43, 0x00, 0x50, 0x00, 0x2d, 0x00, 0x31, 0x00, 0x30, 0x00,
  0x30, 0x00, 0x30, 0x00, 0x20, 0x00, 0x55, 0x00, 0x53, 0x00, 0x42, 0x00, 0x20, 0x00,
  0x61, 0x62, 0x63, 0x64
};
```

Remote Heap Leak (CVE-2021-38665)

Heap Leak

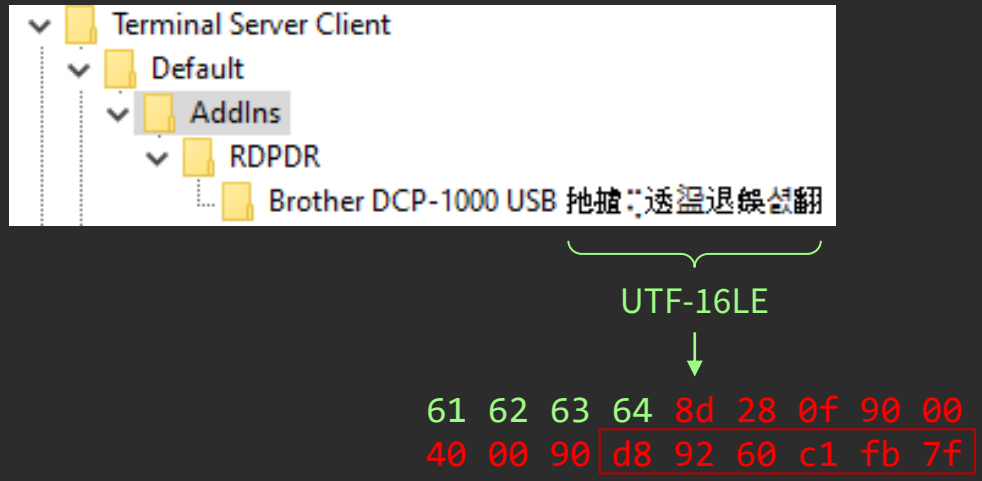
- `W32DrPRN::AddPrinterCacheInfo`
- Juste après l'appel à `RegCreateKeyExW`...



Remote Heap Leak (CVE-2021-38665)

Heap Leak

- `W32DrPRN::AddPrinterCacheInfo`
- Juste après l'appel à `RegCreateKeyExW`...



Remote Heap Leak (CVE-2021-38665)

- Rapatrier le leak

Remote Heap Leak (CVE-2021-38665)

Rapatrier le leak

- Le client envoie un PDU « Device Announce » au démarrage
- Celui-ci contient tous les noms de clés (donc nos leaks !!)

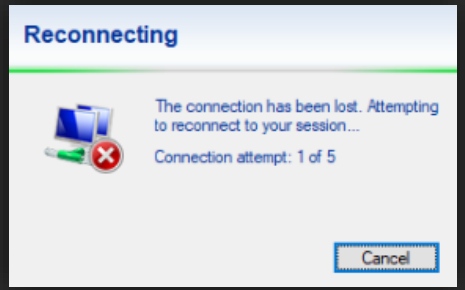
- Peut-on forcer le client à se reconnecter ?

Remote Heap Leak (CVE-2021-38665)

Rapatrier le leak

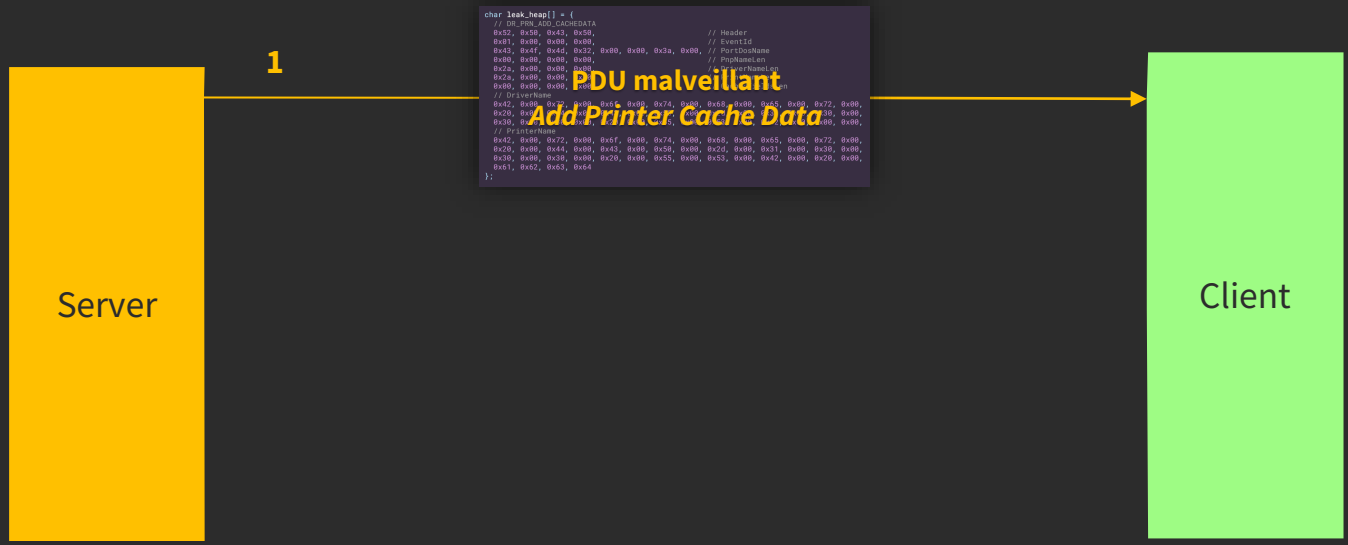
- Le client envoie un PDU « Device Announce » au démarrage
- Celui-ci contient tous les noms de clés (donc nos leaks !!)

- Peut-on forcer le client à se reconnecter ?
  - Corruption du channel `Microsoft::Windows::RDS::Graphics`



Remote Heap Leak (CVE-2021-38665)

Exploit

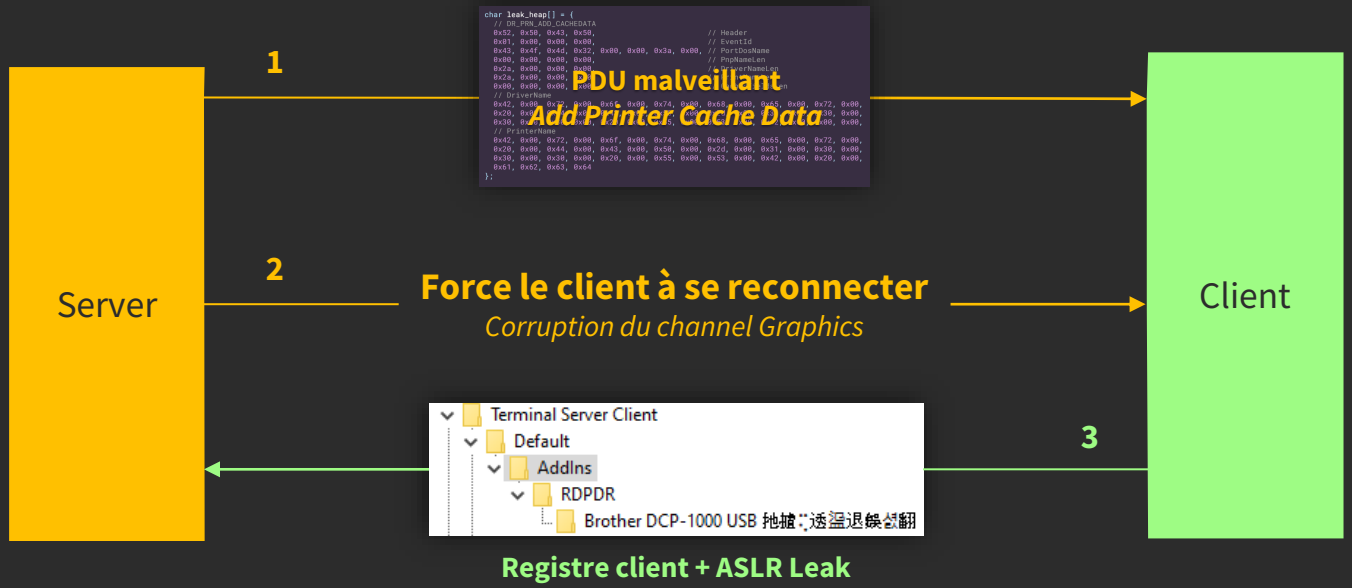


Remote Heap Leak (CVE-2021-38665)

Exploit

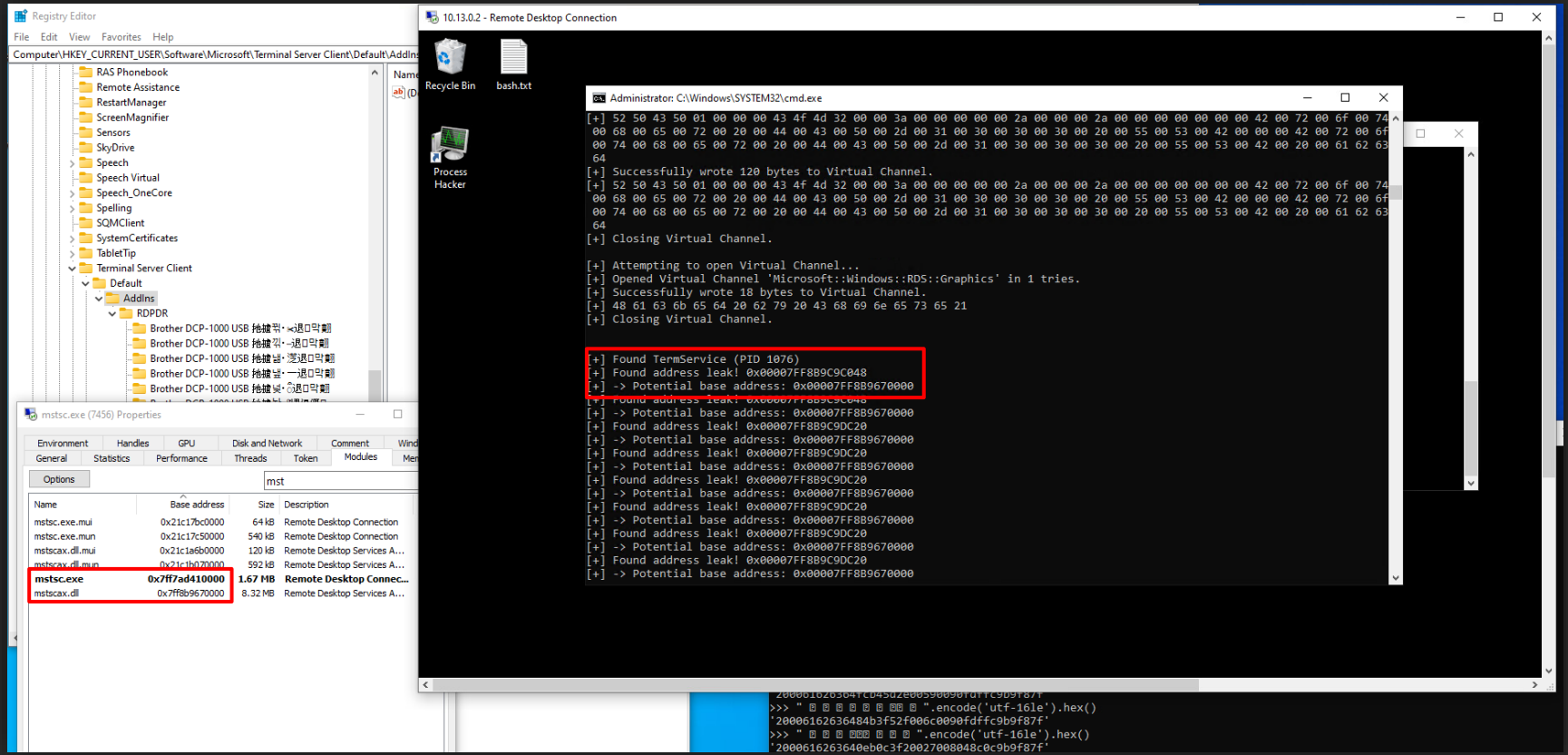
Remote Heap Leak (CVE-2021-38665)

Exploit



Remote Heap Leak (CVE-2021-38665)

Exploit (PoC)



Remote Heap Buffer Overflow (CVE-2021-38666)

Découverte du bug

- Des crashs (!)

Remote Heap Buffer Overflow (CVE-2021-38666)

Découverte du bug

- Des crashs (beaucoup…)

```
Crash at time 1621526903
Exception Address: 00007ff81640d626 / 000000000005d626 (RPCRT4.dll)
Exception Information: 0000000000000000 000002513a542000

Crash at time 1621527242
Exception Address: 00007ff80a777b18 / 0000000000007b18 (WINSPOOL.DRV)
Exception Information: 0000000000000000 000002541565df98

Crash at time 1621527495
Exception Address: 00007ff8166043d2 / 00000000000743d2 (msvcrt.dll)
Exception Information: 0000000000000000 0000023744ced000

Crash at time 1621527745
Exception Address: 00007ff80a774340 / 0000000000004340 (WINSPOOL.DRV)
Exception Information: 0000000000000000 000001835bcc5ee8

Crash at time 1621527779
Exception Address: 00007ff816481bff / 00000000000d1bff (RPCRT4.dll)
Exception Information: 0000000000000000 000002d25e2f2000

Crash at time 1621527853
Exception Address: 00007fffe9a098dc / 000000001cdf98dc (unknown module)
Exception Information: 0000000000000000 00007fffe9a098dc

Crash at time 1621528182
Exception Address: 00007ff80a774340 / 0000000000004340 (WINSPOOL.DRV)
Exception Information: 0000000000000000 000002358eec6ee8

Crash at time 1621528428
Exception Address: 00007ff80a777b18 / 0000000000007b18 (WINSPOOL.DRV)
Exception Information: 0000000000000000 000001cd8e07af98

Crash at time 1621528537
Exception Address: 0000000000000000 / ffff8000333f0000 (unknown module)
Exception Information: 0000000000000000 0000000000000000
```

Remote Heap Buffer Overflow (CVE-2021-38666)

Analyse du crash dans RPCRT4

- `RPCRT4.DLL` : API pour les *Remote Procedure Calls*
- Crash reproductible
- `NdrSimpleTypeConvert`

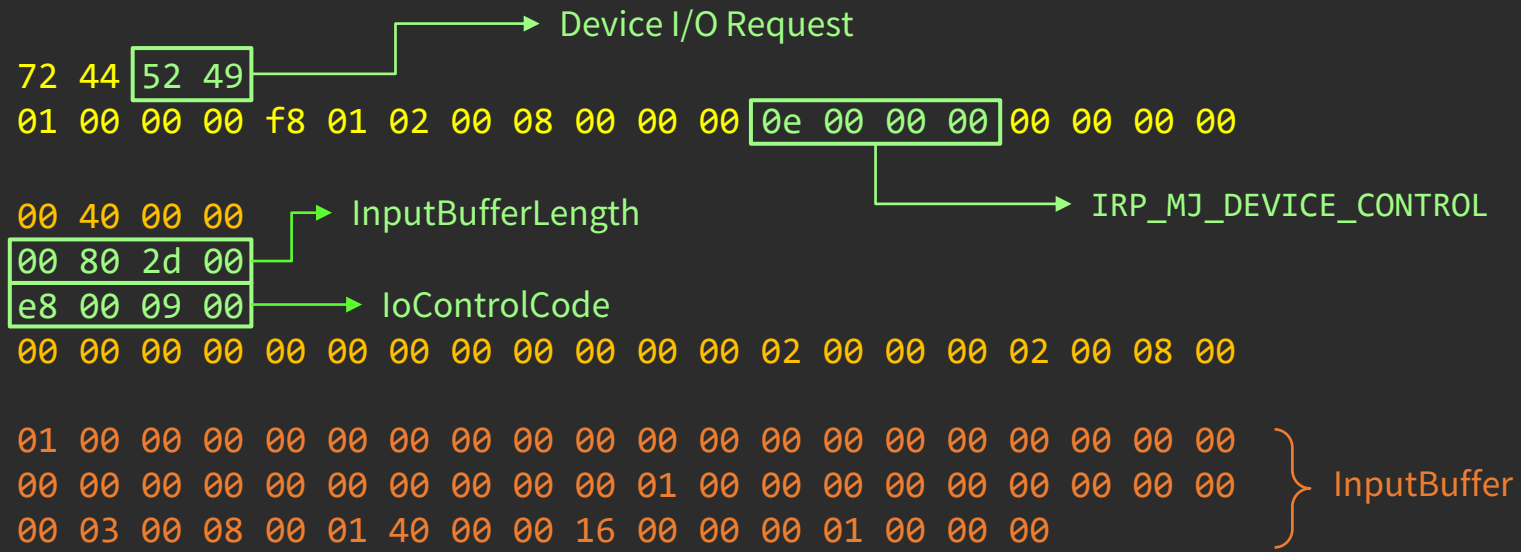
```
mov     eax, [rdx] ; crash
bswap   eax
mov     [rdx], eax
```

Remote Heap Buffer Overflow (CVE-2021-38666)

Analyse du crash dans RPCRT4

- `RPCRT4.DLL` : API pour les *Remote Procedure Calls*
- Crash reproductible
- `NdrSimpleTypeConvert`

```
mov     eax, [rdx] ; crash
bswap   eax
mov     [rdx], eax
```



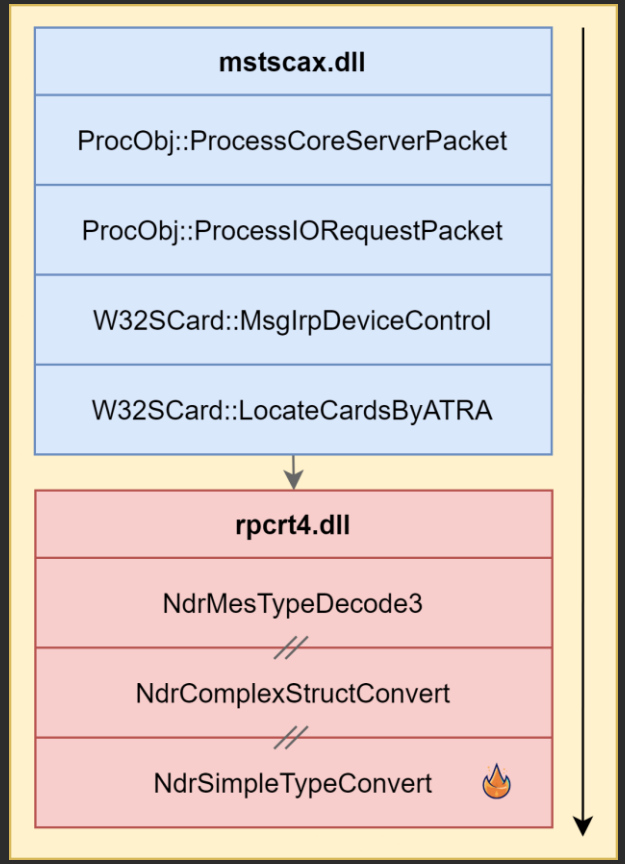
Remote Heap Buffer Overflow (CVE-2021-38666)

55	0x000900DC	SCARD_IOCTL_SETATTRIB	SetAttrib_Call (section 2.2.2.22), Long_Return (section 2.2.3.3)
56	0x000900E0	SCARD_IOCTL_ACCESSSTARTEDEVENT	ScardAccessStartedEvent_Call (section 2.2.2.30), Long_Return (section 2.2.3.3)
57	0x000900E4	SCARD_IOCTL_RELEASETARTEDEVENT	Not used.
58	0x000900E8	SCARD_IOCTL_LOCATECARDSBYATRA	LocateCardsByATRA_Call (section 2.2.2.23), LocateCards_Return (section 2.2.3.5)
59	0x000900EC	SCARD_IOCTL_LOCATECARDSBYATRW	LocateCardsByATRW_Call (section 2.2.2.2

```
72 44 52 49
01 00 00 00 f8 01 02 00 08 00 00 00 0e 00 00 00 00 00 00 00

00 40 00 00      → InputBufferLength
00 80 2d 00
e8 00 09 00      → IoControlCode
00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 02 00 08 00

01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00   } InputBuffer
00 03 00 08 00 01 40 00 00 16 00 00 00 01 00 00 00
```

Remote Heap Buffer Overflow (CVE-2021-38666)

Call stack

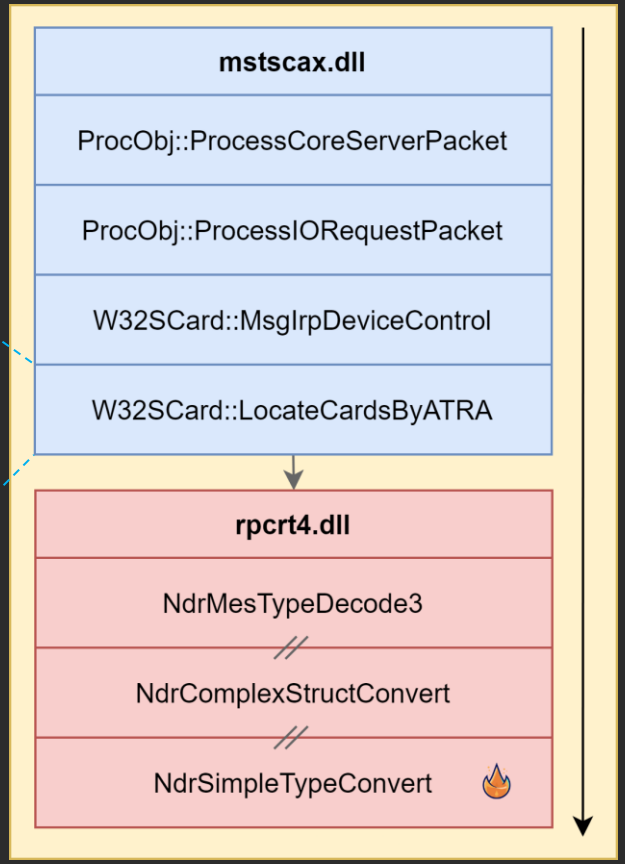


Remote Heap Buffer Overflow (CVE-2021-38666)

Call stack

Pas vérifié !

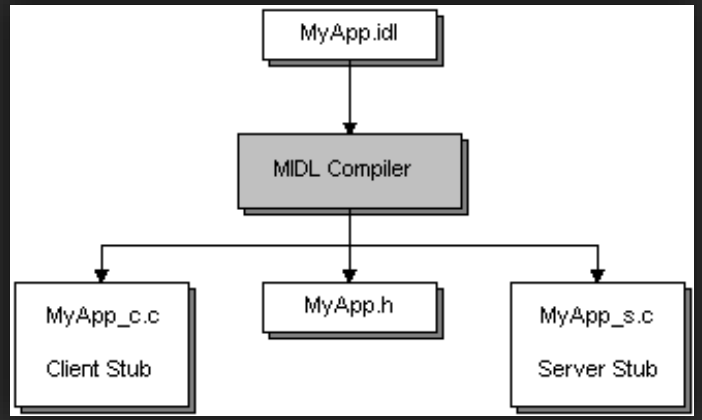
```
MesDecodeBufferHandleCreate(
    &PDU->InputBuffer,
    PDU->InputBufferLength,
    &pHandle
);

// ...

NdrMesTypeDecode3(
    pHandle,
    &pPicklingInfo,
    &pProxyInfo,
    &ArrTypeOffset,
    0xE,
    &pObject
);
```



Remote Heap Buffer Overflow (CVE-2021-38666)

RPC NDR *marshaling engine*

- *Network Data Representation*
- Moteur de sérialisation
- IDL (*Interface Description Language*)

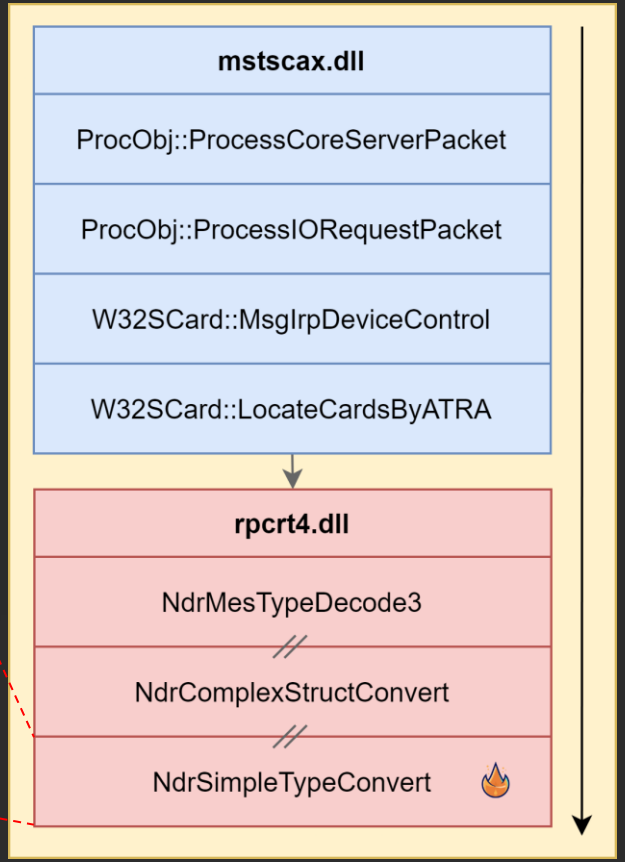



```
MesDecodeBufferHandleCreate(
    &PDU->InputBuffer,
    PDU->InputBufferLength,
    &pHandle
);

// ...

NdrMesTypeDecode3(
    pHandle,
    &pPicklingInfo,
    &pProxyInfo,
    &ArrTypeOffset,
    0xE,
    &pObject
);
```

Remote Heap Buffer Overflow (CVE-2021-38666)

Pré-traitement lors de la désérialisation

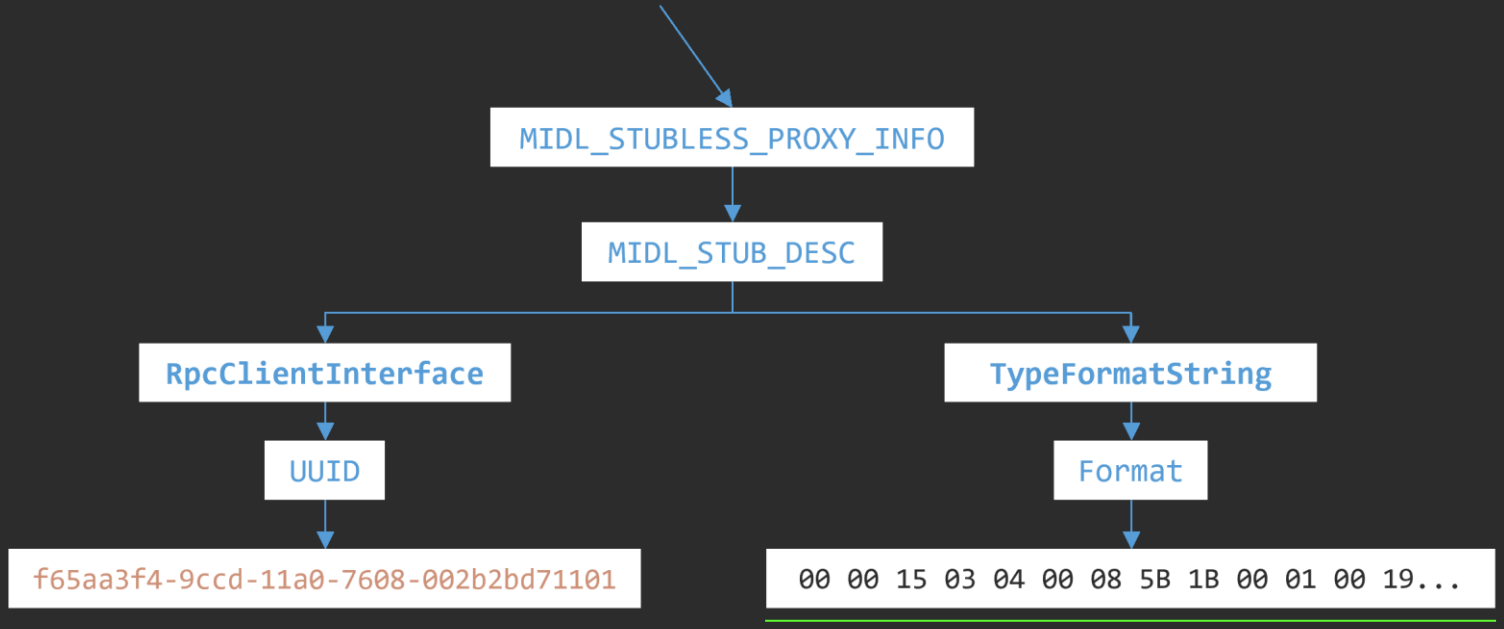
```
void NdrSimpleTypeConvert(PMIDL_STUB_MESSAGE StubMsg, uchar Format) {
    switch (Format) {
        // ...
        case FC_ULONG:
            if ((StubMsg->RpcMsg->DataRepresentation & NDR_INT_REP_MASK) != NDR_LOCAL_ENDIAN) {
                // Crash
                *((ulong *)StubMsg->Buffer) = RtlUlongByteSwap(*(ulong *)StubMsg->Buffer);
            }
            StubMsg->Buffer += 4;
        // ...
    }
}
```



Remote Heap Buffer Overflow (CVE-2021-38666)

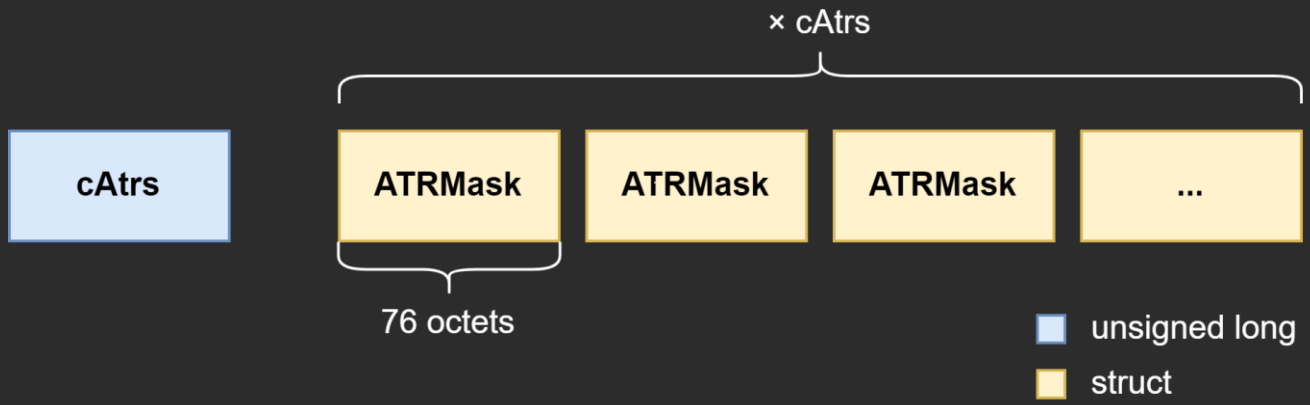
Structure de l'appel LocateCardsByATRA

```
v6 = MesDecodeBufferHandleCreate(&PDU->InputBuffer, PDU->InputBufferLength, &pHandle);
NdrMesTypeDecode3(pHandle, &pPicklingInfo, &pProxyInfo, (const unsigned int **)&ArrTypeOffset, 0xEu, &pObject);
```



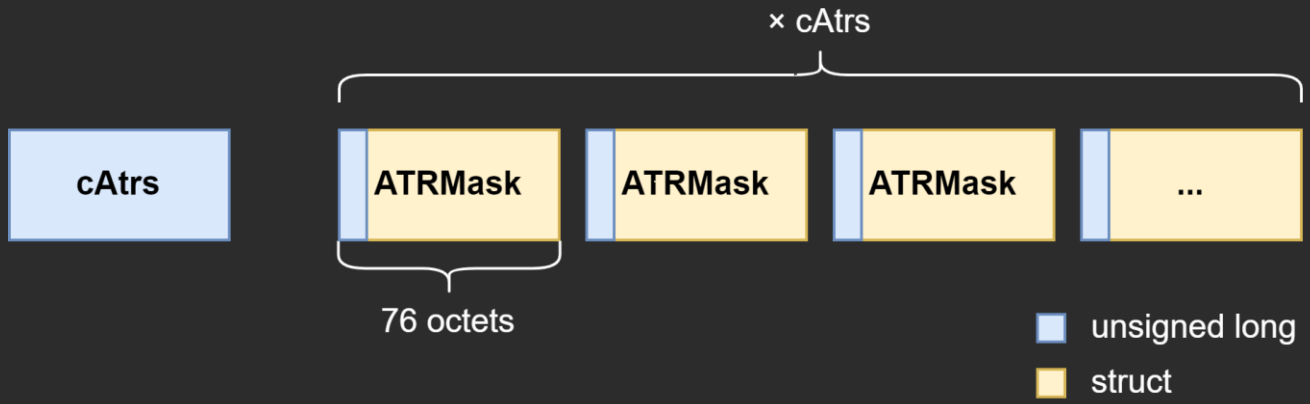
Remote Heap Buffer Overflow (CVE-2021-38666)

Structure de l'appel `LocateCardsByATRA`



Remote Heap Buffer Overflow (CVE-2021-38666)

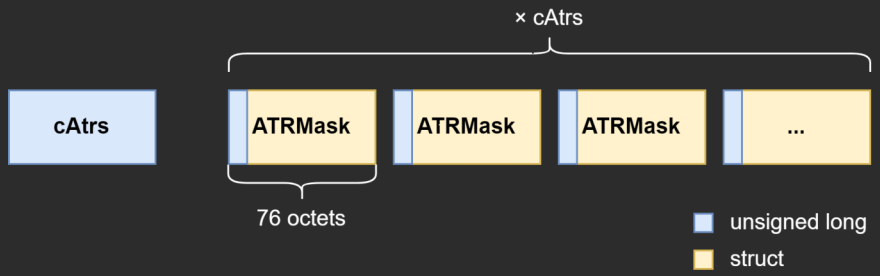
Structure de l’appel `LocateCardsByATRA`



Remote Heap Buffer Overflow (CVE-2021-38666)

Heap Overflow

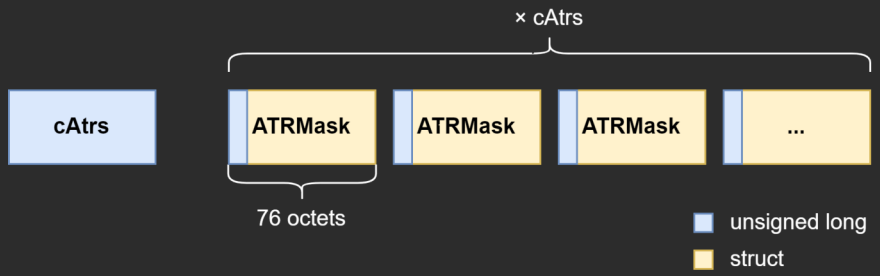
- Combinaison de deux *overflows*
  - InputBufferLength trop grand
  - Contrôle de l'attribut cAtrs dans la structure
- *Byteswap* dans le tas tous les 76 octets



Remote Heap Buffer Overflow (CVE-2021-38666)

Heap Overflow

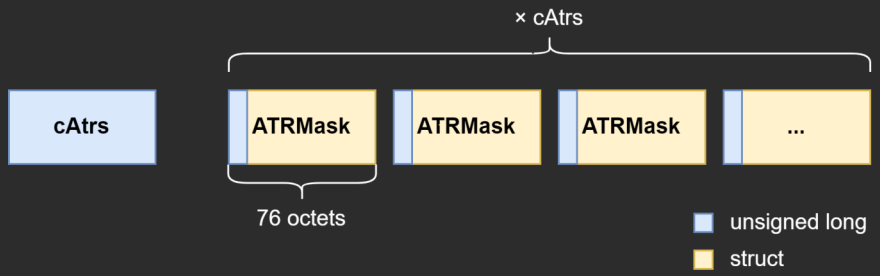
- Combinaison de deux *overflows*
  - InputBufferLength trop grand
  - Contrôle de l'attribut cAtrs dans la structure
- *Byteswap* dans le tas tous les 76 octets
- 62 IOCTLs « Smart Card »
  - Seulement 3 IOCTLs vulnérables
- Corruption du tas, de pointeurs de *vtables*…



Remote Heap Buffer Overflow (CVE-2021-38666)

Heap Overflow

- Combinaison de deux *overflows*
  - InputBufferLength trop grand
  - Contrôle de l'attribut cAtrs dans la structure
- *Byteswap* dans le tas tous les 76 octets
- 62 IOCTLs « Smart Card »
  - Seulement 3 IOCTLs vulnérables
- Corruption du tas, de pointeurs de *vtables*…



Exploit

- Pas de PoC… mais $5000 de bounty
- Explication détaillée + analyse de risque = suffisant !

Conclusion

Conclusion

Résultats

- Construction d'un fuzzer architecturé autour de WinAFL
- Développement d'un harnais qui permet de fuzzer des clients RDP
- Mise en place de différentes stratégies d'attaque

- **Vulnérabilités dans le client RDP de Microsoft**
  - CVE-2021-38665 dans l'extension *Printer*
  - CVE-2021-38666 dans l'extension *Smart Card*

- Réutilisation du fuzzer pour cibler d'autres clients RDP
- **Vulnérabilités dans le client FreeRDP**
  - CVE-2021-37594 (*Remote Memory Leak* dans l'extension presse-papiers)
  - CVE-2021-37595 (*Remote Arbitrary File Read* dans l'extension presse-papiers)

Merci !

Questions ?

https://thalium.re/