Holding the Stick

RDPFuzz

At Both Ends

Or Ben-Porath & Shaked Reiner

CyberArk Labs
Who are we

Shaked Reiner
Principal Security Researcher
@ShakReiner

Or Ben-Porath
Security Researcher
@OrBenPorath
Agenda

01 RDP
   How the protocol works

02 Fuzzing RDP
   Our fuzzing process

03 Results
   Fuzzing Stats

04 Summary
   Conclusion and future work
Why RDP? What’s the attack surface? How does it work?
### TOTAL RESULTS

<table>
<thead>
<tr>
<th>Country</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>1,452,039</td>
</tr>
<tr>
<td>China</td>
<td>1,158,306</td>
</tr>
<tr>
<td>Germany</td>
<td>193,084</td>
</tr>
<tr>
<td>Netherlands</td>
<td>126,017</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>118,304</td>
</tr>
</tbody>
</table>

### RDP In the Wild
RDP Attack Vector #1
RDP Attack Vector #2
Examples

DejaBlue
Int overflow
RCE

Reverse RDP
Path traversal in clipboard
Run arbitrary applications
Attack Surface

```powershell

Count
-----
191
```
How Does RDP Work?
Read Surface

PS C:\research\RDP\RDP-SPECS> gci -r *MS-RDP*.pdf | Measure-Object | select count

<table>
<thead>
<tr>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>30</td>
</tr>
</tbody>
</table>
RDP Components

**Server**
- svchost.exe
- termsrv.dll
- rdpbase.dll
- rdpclip.exe
- audiodg.exe
- rdpdr.sys
- rdpbus.sys

**Client**
- mstsc.exe
- mstscax.dll

**RDP**
- mstsc.exe
- mstscax.dll
RDP Components

Server
- termsrv.dll
- rdpbase.dll
- rdpclip.exe
- rdpdr.sys
- svchost.exe

Client
- mstsc.exe
- mstscax.dll
02 RDPFuzz

Fuzzing RDP
Classic Fuzzing
Coverage-guided Fuzzing Setup

Fuzzer → PDF → PDF Reader

Code Coverage
Fuzzing Options

- **Open-source**
  - Use a modified open-source client/server

- **Custom**
  - Write our own client and server

- **Snapshot**
  - Use a snapshot-based fuzzer

- **Use existing**
  - Tap into the client and server using legitimate APIs or code injection
Credit: Fuzzing and Exploiting Virtual Channels in Microsoft Remote Desktop Protocol for Fun and Profit
Background Fuzzing
### 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 tags.

<table>
<thead>
<tr>
<th>msgType</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNDC_CLOSE</td>
<td>Close PDU</td>
</tr>
<tr>
<td>SNDC_WAVE</td>
<td>Wave PDU</td>
</tr>
<tr>
<td>SNDC_SETVOLUME</td>
<td>Volume PDU</td>
</tr>
<tr>
<td>SNDC_SETPITCH</td>
<td>Pitch PDU</td>
</tr>
<tr>
<td>SNDC_WAVECONFIRM</td>
<td>Wave Confirm PDU</td>
</tr>
<tr>
<td>SNDC_TRAINING</td>
<td>Training PDU or Training Confirm PDU</td>
</tr>
<tr>
<td>SNDC_FORMATS</td>
<td>Send Audio Formats and Version PDU or Clear Audio Formats and Version PDU</td>
</tr>
<tr>
<td>SNDC_KEYPTST</td>
<td>Key PTST PDU</td>
</tr>
<tr>
<td>SNDC_WAVECRYPT</td>
<td>Wave Crypt PDU</td>
</tr>
<tr>
<td>SNDC_UDPFUTURE</td>
<td>UDP PDU</td>
</tr>
<tr>
<td>SNDC_UDPFUTURELAST</td>
<td>UDP Last PDU</td>
</tr>
<tr>
<td>SNDC_QMVarious</td>
<td>Quality Mode PDU</td>
</tr>
<tr>
<td>SNDC_WAVE2</td>
<td>Wave2 PDU</td>
</tr>
</tbody>
</table>
Statefulness
Statefulness

Client

CLOSED

Server
Statefulness

Code patches

Grammar enforcement
Code Patches

mstscax!RdpGfxClientChannel::OnDataReceived

loc_16A7C1276:
cmp    dword ptr [rdi+68h], 0
jz     loc_16A7C1382

loc_16A7C12C4:
    ; this
    lea   rcx, [rdi-30h]
    mov   edx, esi
    ; int
    call  jRdpGfxClientChannel1@EAA93BC ; RdpGfxClientChannel::ReconnectSoftwareMode(long)
    mov   esi, eax
    test  eax, eax
    jns    short loc_16A7C1313
drwrap_replace()

DR_EXPORT bool drwrap_replace (app_pc original,
               app_pc replacement,
               bool override

)

Replaces the application function that starts at the address original with the code at the address replacement.

Only one replacement is supported per target address. If a replacement already exists for original, this function fails unless override is true, in which case it replaces the prior replacement. To remove a replacement, pass NULL for replacement and true for override. When removing or replacing a prior replacement, existing replaced code in the code cache will be flushed lazily. i.e., there may be some execution in other threads after this call is made.

Only the first target replacement address in a basic block will be honored. All code after that address is removed.

When replacing a function, it is up to the user to ensure that the replacement mirrors the calling convention and other semantics of the original function. The replacement code will be executed as application code, NOT as client code.

Note

The priority of the app2app pass used here is DRMGR_PRIORITY_APP2APP_DRWRAP and its name is DRMGR_PRIORITY_NAME_DRWRAP.

Returns

whether successful.
Grammar Enforcement

Documentation
Reading the docs

RE
Analyzing the conditions within the code

Tracing
Analyzing failed executions
Grammar EnFORCEment

Limit Fuzzer

Grammar narrows down the input space
Multi-input

Client  Test Case  Test Case  Test Case  Test Case  Test Case  Server
Multi-input
### 2.2.2.9 RDPGFX_CREATE_SURFACE_PDU

04/07/2021 • 2 minutes to read

The **RDPGFX_CREATE_SURFACE_PDU** message is used to instruct the client to create a surface of a given width, height, and pixel format.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>1234567890123456789012345678901</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>surfaceId</td>
<td>width</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>pixelFormat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.2.2.15 RDPGFX_MAP_SURFACE_TO_OUTPUT_PDU

02/14/2019 • 2 minutes to read

The **RDPGFX_MAP_SURFACE_TO_OUTPUT_PDU** message is sent by the server to instruct the client to map a surface to a rectangular area of the Graphics Output Buffer (section 3.3.1.7) ADM element.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>1234567890123456789012345678901</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>surfaceId</td>
<td>reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>outputOriginX</td>
<td>outputOriginY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Test Case

- cmd07 <1st PDU data>
- cmd02 <2nd PDU data>
- cmd03 <3rd PDU data>
Multi-channel Input
Video::Control

Video::Data
__cmd07 <1st PDU data> __cmd02 <2nd PDU data> __cmd03 <3rd PDU data>
Locating Target Functions

+ Too many components

+ PDBDownloader to download all relevant .pdb files

+ grep/sls to get all the C<\text{class-name}>\text{::OnDataReceived} functions
A FEW FUZZING ITERATIONS LATER
Reproduction Issues

:)  

Sorry, no crash
Automatic Crash Analysis

watchdog.ps1

mstsc.exe

Stacks

DUMP  DUMP  DUMP
DUMP  DUMP  DUMP
DUMP  DUMP  DUMP
DUMP  DUMP  DUMP
From: rdpfuzzmonitor
Subject: NEW CRASH in AS_SERVER4 :)

Body:

mstscax!CRdpAudioController::OnWaveData+0x281
mstscax!CRdpAudioController::DataArrived+0x72f
mstscax!CRdpAudioPlaybackChannelCallback::OnDataReceived+0x433
mstscax!CDynVCChannel::InvokeCallback+0x1b0
mstscax!CDynVCChannel::OnData+0x3aa
mstscax!CDynVCPlugin::OnStaticDataReceived+0x14f
mstscax!CStaticChannelCallback::OnDataReceived+0x24
mstscax!CCommonVCChannel::OpenProcEx+0x31e
mstscax!CCommonVCChannel::static_OpenProcEx+0xc6
mstscax!CChan::ChannelOnPacketReceived+0x179
mstscax!CSSL::SLReceivedDataPacket+0x110
mstscax!CSSL::OnPacketReceived+0x19d
mstscax!CMCS::MCSRecvData+0x28f
mstscax!CMCS::OnDataAvailable+0xdd
mstscax!CTSC224Filter::OnDataAvailable+0x138
mstscax!CTscSs1Filter::OnDataAvailable+0xda
mstscax!CTSCFilterTransport::OnDataAvailable_TransportEvent+0x63
mstscax!CSTTransportStack::OnDataAvailable+0x106
mstscax!CSTTcpTransport::AsyncOnReadCompletedAsyncCallback::Invoke+0x60
mstscax!CSTMsg::Invoke+0xdc
mstscax!CSTThread::RunAllQueueEvents+0x219
mstscax!CSTThread::internalMsgPump+0x91
mstscax!CSTThread::internalThreadMsgLoop+0x14d
mstscax!CSTThread::ThreadMsgLoop+0x1c
mstscax!CRCV::RCVMain+0x170
mstscax!CSTThread::TSStaticThreadEntry+0x258
mstscax!PAL_System_Win32_ThreadProcWrapper+0x32
KERNEL32!BaseThreadInitThunk+0x14
ntdll!RtlUserThreadStart+0x21
03 Results
Summary stats
============

- Fuzzers alive: 10
- Total run time: 0 days, 3 hours
  - Total execs: 0 million
- Cumulative speed: 41.39 execs/sec
- Pending paths: 154 faves, 664 total
- Maximal coverage: 3.84%
- Average coverage: 3.01%
- Average stability: 15.56%
- Pending per fuzzer: 15.40 faves, 66.40 total (on average)
- Crashes found: 17 locally unique
~1 month
Fuzzing duration

5
bugs

15
Channels fuzzed
AUDIO_PLAYBACK Channel

RDP Server → AUDIO_PLAYBACK Channel → RDP Client
- offset -  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0x00000000  5f5f  5f63  6d64  3037  6f02  0000  90d8  deeb
0x00000010  6f02  0000  270c  1a00  ff08  00b2  06a1  0200
0x00000020  5f5f  5f63  6d64  3064  0000  0500  00ff  0800
0x00000030  b206  a102  0044  ac00  00c0  5d00  0004  0010
0x00000040  0000  0006  a17e  1100  5f5f  5f63  6d64  3037
0x00000050  6f00  0000  06a1  0200  44ac  0000  0000  0200
0x00000060  4001  0200  0100  401f  0000  0010  0000  803e
0x00000070  5f5f  5f63  6d64  3064  0000  0500  00ff  0800
0x00000080  b206  a102  0044  ac00  00c0  5d00  0004  0010
0x00000090  0000  0006  a17e  1100

0123456789ABCDEF
___cmd07o........
___cmd0d........
.....D....]
.....~.___cmd07
o.......D.....
@.....@.......>
___cmd0d........
.....D....]
.....~....
// mstscax!CRdpAudioController::OnWaveData
last_format = pThis->last_format;
format_from_msg = *((unsigned __int16*)msg + 3);

if (last_format != format_from_msg)
{
    /* ... */
    // Treat format change
    v5 = CRdpAudioController::OnNewFormat(pThis, (__int64*)format_from_msg);
    /* ... */
    pThis->last_format = last_format = format_from_msg;
}

formats_array = (AUDIO_FORMAT**)pThis->formatArray;
/* ... */
current_wFormatTag = formats_array[last_format]->wFormatTag;
RDP Server

type: formats
NumFormats: 3

Format Array

RDP Client
RDP Server

type: formats
NumFormats: 3

RDP Client

Format Array

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>

RDP Server

type: data
format: 3
00 11 22 33
22 33 44 55
**RDP Server**

```
<table>
<thead>
<tr>
<th>type: formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumFormats: 3</td>
</tr>
</tbody>
</table>
```

**RDP Client**

```
| type: data |
| format: 3 |
| 00 11 22 33 |
| 22 33 44 55 |
```

Format Array:

```
[ unused ]
```
RDP Server

```
type: formats
NumFormats: 3
```

```
type: data
format: 3
  00 11 22 33
  22 33 44 55
```

RDP Client

```
type: formats
NumFormats: 1
```

Format Array

```
[ ] [ ] [ ]
```

RDP Client
RDP Server

**type:** formats  
**NumFormats:** 3

**type:** data  
**format:** 3  
```
00 11 22 33  
22 33 44 55
```

RDP Client

**type:** formats  
**NumFormats:** 1

**type:** data  
**format:** 3  
```
00 11 22 33  
22 33 44 55
```
// mstscax!CRdpAudioController::OnWaveData
last_format = pThis->last_format;
format_from_msg = *((unsigned __int16*)msg + 3);

if (last_format != format_from_msg)
{
    /* ... */
    // Treat format change
    v5 = CRdpAudioController::OnNewFormat(pThis, (__int64*)format_from_msg);
    /* ... */
    pThis->last_format = last_format = format_from_msg;
}

formats_array = (AUDIO_FORMAT**)pThis->formatArray;
/* ... */
current_wFormatTag = formats_array[last_format]->wFormatTag;
FUZZ

ALL THE THINGS
Future Work

RDP
RPC
R**

https://github.com/cyberark/RDPFuzz
THANKS

@OrBenPorath
@ShakReiner

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik