

Fuzzing Remote Desktop Protocol

RDP에 대한 이해부터 퍼징, 취약점 발굴까지

2023.7.3

발표자: 최지현

with Team BTS (BoB 11th)

Team Introduction



멘토

박세한



최지현 (PM)
프로젝트 총괄



김경민
퍼저 제작, 취약점 분석



송준현
익스플로잇, 취약점 분석



PL

정수환



조성준
퍼저 설계, 취약점 분석



채하늘
리버싱, 퍼저 제작

Speaker Introduction



최지현

팀 Project Manager 수행

BoB 11기 보안 컨설팅 트랙 BEST 10

Table of Contents

01. RDP 개요

02. RDP 프로토콜 분석

03. RDP 취약점 발굴

04. 결론



RDP 개요

What is RDP?

Microsoft사에서 원격 데스크톱 연결을 위해 개발한 프로토콜



[MS-RDPBCGR]:
Remote Desktop Protocol: Basic Connectivity and Graphics Remoting

[MS-RDPCR2]:
Remote Desktop Protocol: Compositing Remoting V2

[MS-RDPECLIP]:
Remote Desktop Protocol: Clipboard Virtual Channel Extension

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting ipig@microsoft.com.
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

What are RDP services?

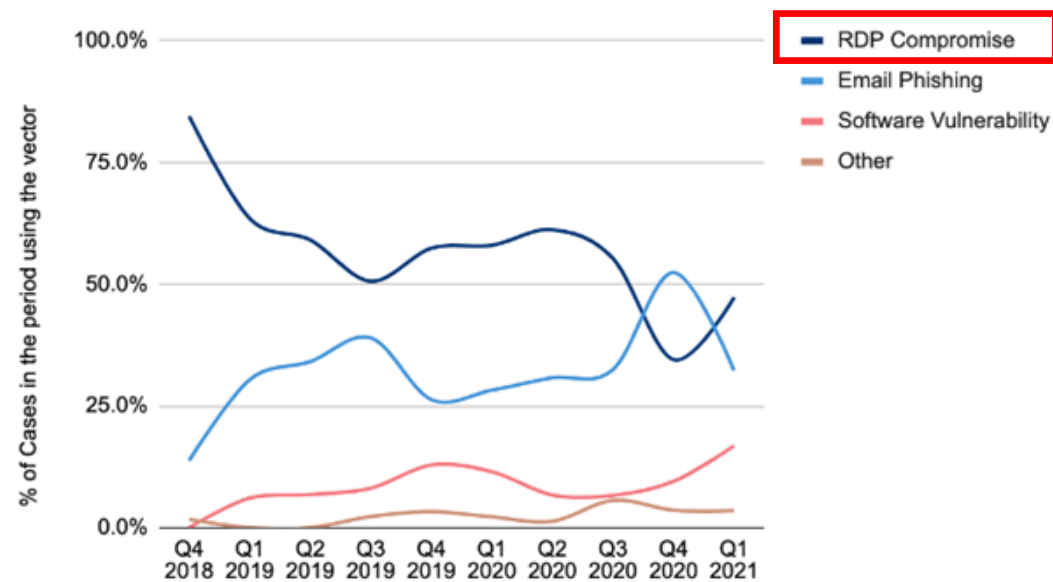


다양한 환경의 Client/Server 사이드 RDP 서비스 존재

Why RDP?

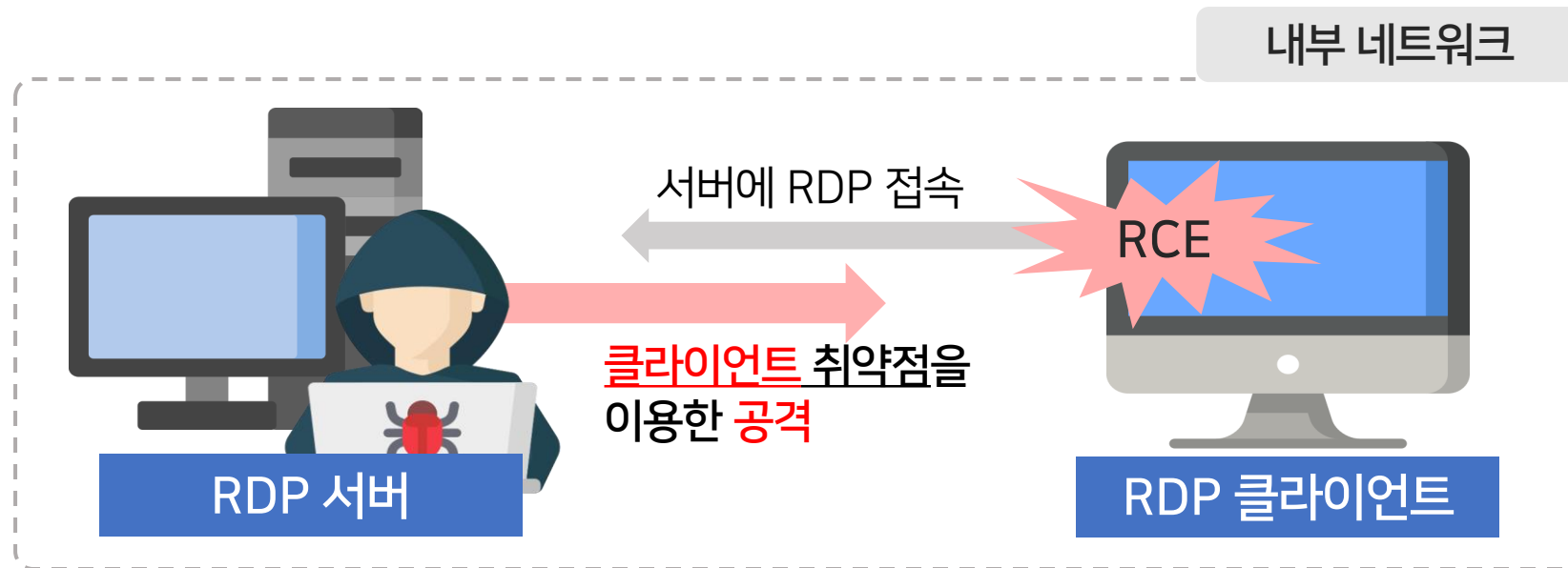
Lateral Movement

| | | | |
|--|-------|--------------------------------------|-------|
| T1021: Remote Services | 27.4% | T1021.001: Remote Desktop Protocol | 23.4% |
| | | T1021.004: SSH | 4.6% |
| | | T1021.002: SMB/Windows Admin Shares | 4.0% |
| | | T1021.005: VNC | 0.5% |
| | | T1021.006: Windows Remote Management | 0.2% |
| T1550: Use Alternate Authentication Material | 0.8% | T1550.002: Pass the Hash | 0.5% |
| | | T1550.001: Application Access Token | 0.2% |
| | | T1550.003: Pass the Ticket | 0.2% |
| T1570: Lateral Tool Transfer | 0.6% | | |
| T1534: Internal Spearphishing | 0.5% | | |



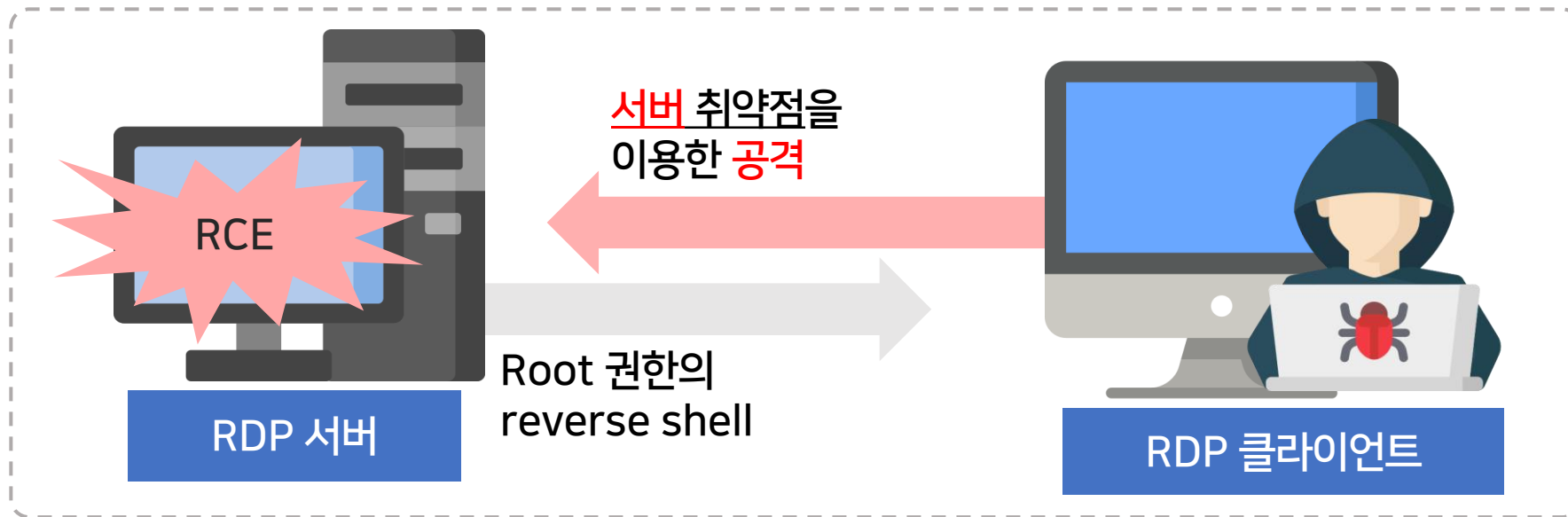
Possible Scenarios

Attack vector 1 : Client

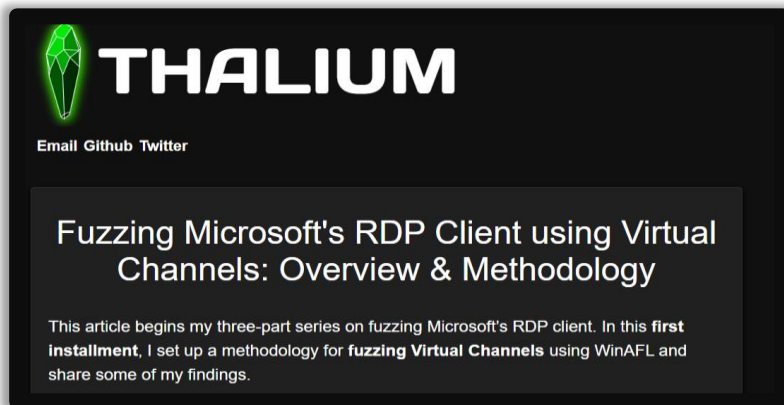


Possible Scenarios

Attack vector 2 : Server (post-auth)



Previous Research



THALIUM
Email Github Twitter

Fuzzing Microsoft's RDP Client using Virtual Channels: Overview & Methodology

This article begins my three-part series on fuzzing Microsoft's RDP client. In this **first installment**, I set up a methodology for **fuzzing Virtual Channels** using WinAFL and share some of my findings.



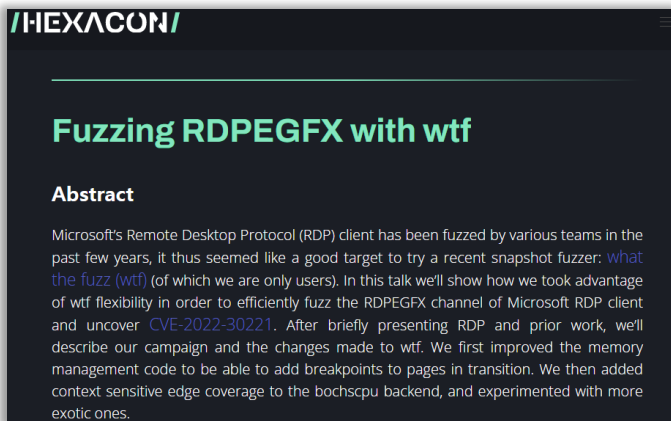
Resource Menu

All > Threat Research Blog > Fuzzing RDP: Holding the Stick at Both Ends

Fuzzing RDP: Holding the Stick at Both Ends

Shaked Reiner And Or Ben-Porath | 8/27/21

Share This!    



HEXACON

Fuzzing RDPE GFX with wtf

Abstract

Microsoft's Remote Desktop Protocol (RDP) client has been fuzzed by various teams in the past few years, it thus seemed like a good target to try a recent snapshot fuzzer: *what the fuzz (wtf)* (of which we are only users). In this talk we'll show how we took advantage of wtf flexibility in order to efficiently fuzz the RDPE GFX channel of Microsoft RDP client and uncover [CVE-2022-30221](#). After briefly presenting RDP and prior work, we'll describe our campaign and the changes made to wtf. We first improved the memory management code to be able to add breakpoints to pages in transition. We then added context sensitive edge coverage to the bochscpu backend, and experimented with more exotic ones.

Fuzzing and Exploiting Virtual Channels in Microsoft Remote Desktop Protocol for Fun and Profit

[Chun Sung Park](#) | Graduate Student, Korea University

[Yeongjin Jang](#) | Assistant Professor, Oregon State University



[Seungjoo Kim](#) | Professor, Korea University

[Ki Taek Lee](#) | PhD Candidate / Principal Engineer, Korea University / Samsung Research

Location: Room B

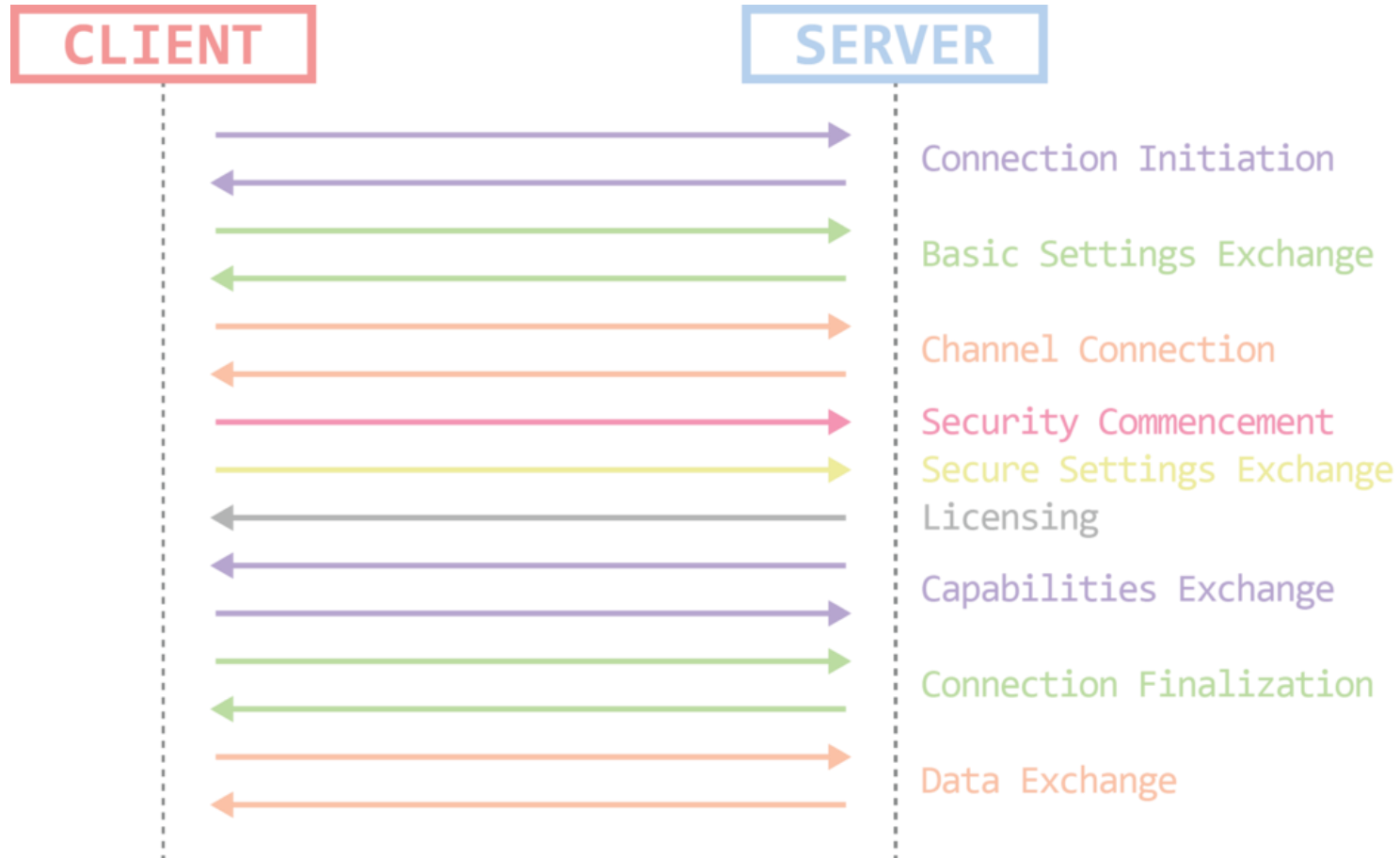
Date: Wednesday, December 4 | 4:50pm–5:40pm

Format: 50–Minute Briefings

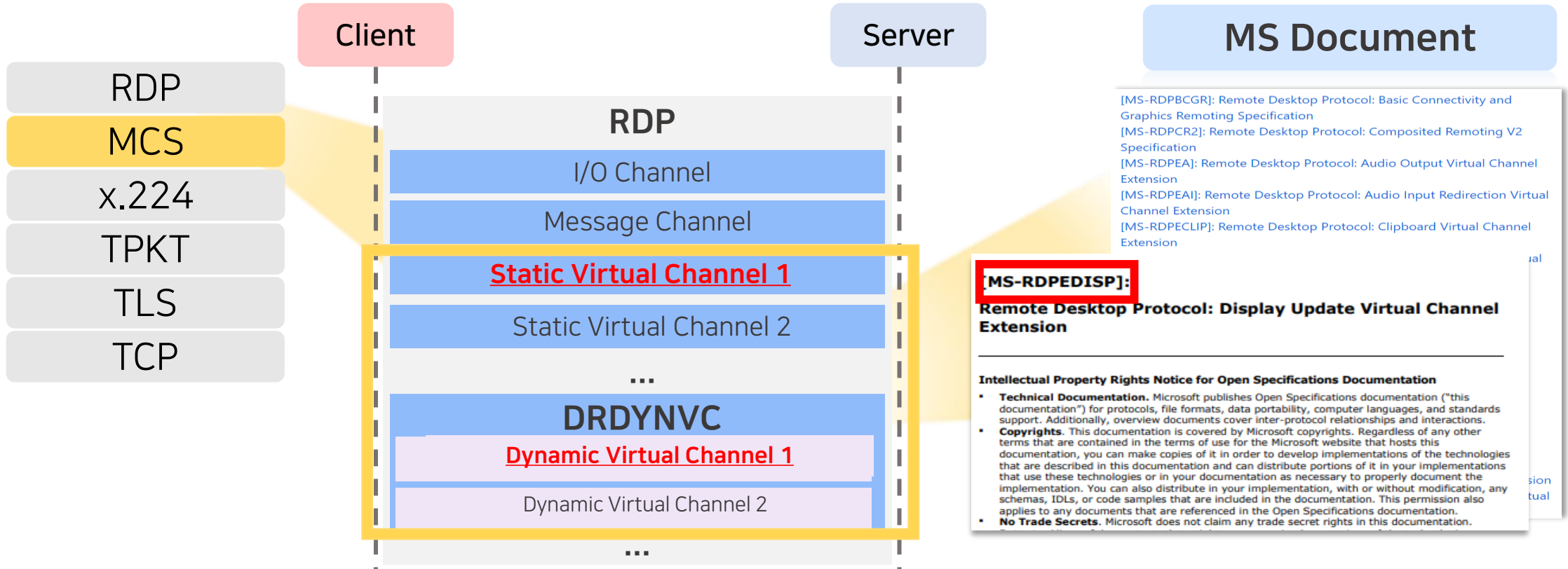
Tracks:  Platform Security,  Exploit Development

RDP 프로토콜 분석

RDP Connection



Protocol Analysis



약 33개의 관련 문서 존재

RDP Channel Analysis

| | | | |
|--|--------------------------------|---------|--------------------|
| RDPEXPS  OPEN | XML Paper Specification: Print | Dynamic | Device Redirection |
| RDPCR2 | Composited Remoting V2 | Dynamic | Graphics |
|  RDPEPNP | Plug and Play | Dynamic | Device Redirection |
|  RDPEAI | Audio Input | Dynamic | Data Redirection |
|  RDPEV | Video | Dynamic | Data Redirection |
| RDPEUSB | USB Devices | Dynamic | Device Redirection |
|  RDPEGFX | Graphic Pipeline | Dynamic | Graphics |
|  RDPEI | Pen & Touch Input | Dynamic | Data Redirection |
|  RDPEVOR | Video Optimized Remoting | Dynamic | Graphics |
| RDPEECO | Echo | Dynamic | Data Redirection |
|  RDPEGT | Geometry Tracking | Dynamic | Graphics |
|  RDPEDISP | Display Control | Dynamic | Graphics |
|  RDPEAR | Authentication Redirection | Dynamic | Authentication |
|  RDPECAM | Camera | Dynamic | Device Redirection |

RDP Channel Analysis (FreeRDP Example)

| 채널명 | 취약점 분석 방법 | 발견된 취약점 | 참조 문서 | 채널 설명 | 선정 이유 |
|-----------------|--------------------|--|---------|----------------------------------|--|
| parallel | 코드 오디팅 | Uninitialized Memory로 인한 Leak 취약점 (총 1개) | RDPEFS | 파일 시스템 지원 | 가장 용이하게 Leak 취약점을 발견함. |
| tsmf | 코드 오디팅 | Heap Buffer Overflow 외 4 (총 5개) | RDPEV | 오디오/비디오를 server에서 client으로 리다이렉션 | 간단한 코드 오디팅을 진행 한 후 가장 취약한 채널이라 판단됨. |
| video | 코드 오디팅, 간단한 fuzzer | Use After free 취약점 외 1 (총 2개) | RDPEVOR | 비디오 stream을 host에서 client로 리다이렉션 | (1) 간단한 코드 오디팅을 진행 한 후 취약한 채널이라 판단됨. (2) 패킷 PDU 구현이 비교적 간단하여 Fuzzer 제작이 용이했음. |
| rdpgfx | AFL++ 기반 fuzzing | Out of Bound Read 취약점 외 1 (총 2개) | RDPGFX | 그래픽 디스플레이 인코딩 | 디코딩이 사용되는 채널로, 취약점이 발생할만한 지점으로 판단됨. |
| geometry | AFL++ 기반 fuzzing | 아직 발견X | RDPEGT | Host와 client 사이의 그래픽 렌더링을 조정함 | 패킷 PDU 구현이 간단해 AFL++ 기반 변형 Fuzzer 구현 연습을 위해 사용함. |
| encomsp | AFL++ 기반 fuzzing | 아직 발견X | RDPEMC | 다수의 커넥션의 참여자들 사이의 메시지 공유 | 패킷 PDU 구현이 간단해 AFL++ 기반 변형 Fuzzer 구현 연습을 위해 사용함. |
| rdpsnd | 간단한 fuzzer | 아직 발견X | RDPSND | 오디오 지원 | 패킷 PDU 구현이 간단해 Fuzzer 제작이 용이했음. |

RDP 취약점 발굴

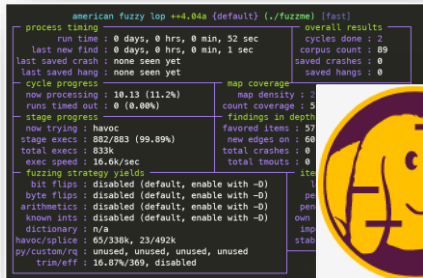
Different Methods



CodeQL 사용

- 1 비슷한 취약점 패턴 탐지 위함
- 2 오픈소스 대상 분석 시 사용

FreeRDP, rdesktop



AFL++ 기반 Fuzzer

- 1 오픈소스 대상 분석 시 사용
- 2 코드 양이 비교적 많은 타겟

FreeRDP, xrdp



Jackalope 기반 Fuzzer

- 1 윈도우, Closed-source
- 2 Output Filtering, Grammar Based Fuzzing

MS RDP

(FreeRDP) Using CodeQL

- 쿼리를 이용해 FreeRDP 및 다른 오픈소스 분석 진행
- 쿼리 실행 및 manual 분석 함께 진행

```

1 import cpp
2 import semmle.code.cpp.rangeanalysis.SimpleRangeAnalysis
3 import semmle.code.cpp.dataflow.TaintTracking
4 import semmle.code.cpp.models.interfaces.DataFlow
5 import semmle.code.cpp.controlflow.Guards
6 import semmle.code.cpp.dataflow.DataFlow
7
8 Quick Evaluation: isUnsafeMalloc
9 predicate isUnsafeMalloc(FunctionCall mallocCall) {
10   exists(Function malloc |
11     malloc.hasQualifiedName("malloc") and
12     mallocCall.getTarget() = malloc and
13     mallocCall.getArgument(0).getType().getSize() != 8 and
14     not mallocCall.getArgument(0).isConstant() and
15     mallocCall.getArgument(0).getNumChild() > 1 and
16     (
17       upperBound(mallocCall.getArgument(0).getFullyConverted()) >= 268435456*16 or
18       lowerBound(mallocCall.getArgument(0).getFullyConverted()) < 0
19     )
20   )
21 }
22 Quick Evaluation: isUnsafeCalloc
23 predicate isUnsafeCalloc(FunctionCall callocCall) {
24   exists(Function calloc |
25     calloc.hasQualifiedName("calloc") and
26     callocCall.getTarget() = calloc and
27     callocCall.getArgument(1).getType().getSize() != 8 and
28     not callocCall.getArgument(1).isConstant() and
29     callocCall.getArgument(1).getNumChild() > 1 and
30     (
31       upperBound(callocCall.getArgument(1).getFullyConverted()) >= 268435456*16 or
32       lowerBound(callocCall.getArgument(1).getFullyConverted()) < 0
33     )
34   )
35 }
36 Quick Evaluation: isUnsafeStreamNew
37 predicate isUnsafeStreamNew(FunctionCall streamNewCall) {
38   exists(Function streamNew |
39     streamNew.hasQualifiedName("Stream_New") and
40     streamNewCall.getTarget() = streamNew and
41     streamNewCall.getArgument(1).getType().getSize() != 8 and
42     not streamNewCall.getArgument(1).isConstant() and
43     streamNewCall.getArgument(1).getNumChild() > 1 and
44     (
45       upperBound(streamNewCall.getArgument(1).getFullyConverted()) >= 268435456*16 or
46       lowerBound(streamNewCall.getArgument(1).getFullyConverted()) < 0
47     )
48   )
49 }

```

Division by Zero

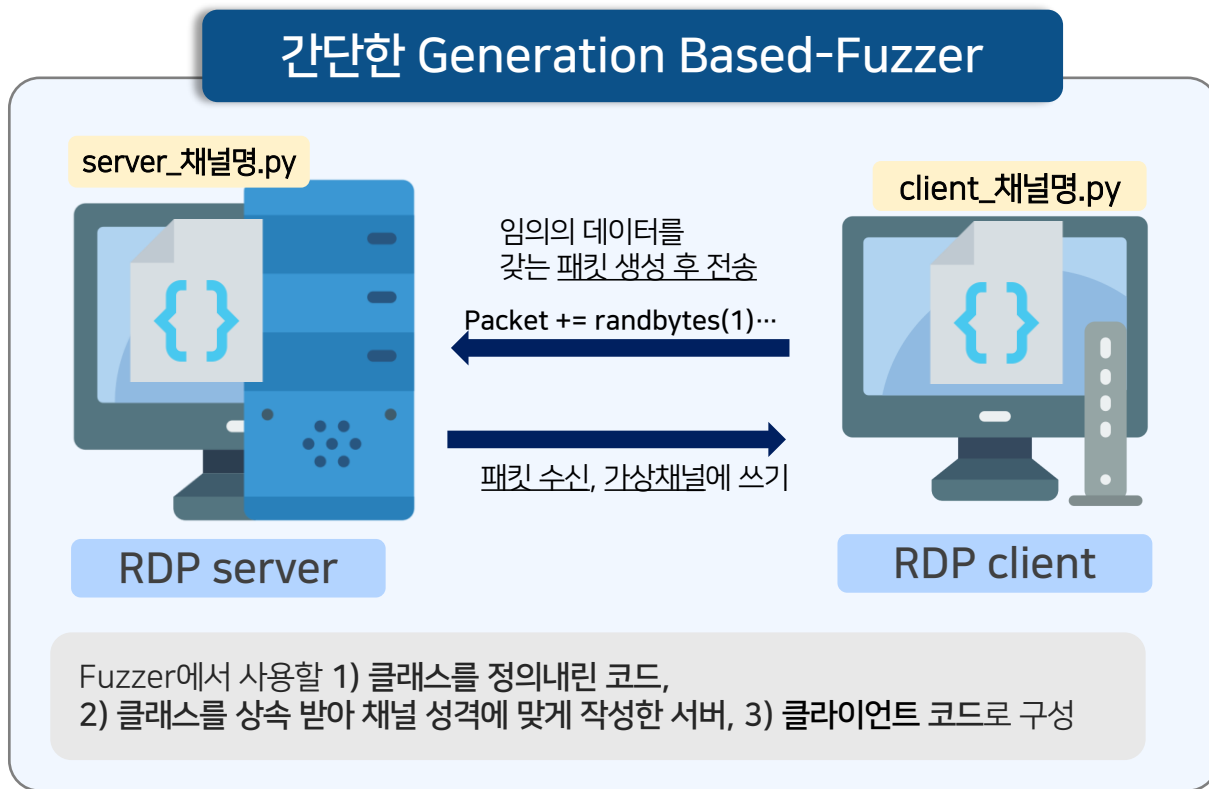
스트림에서 읽어온 값이
Integer Overflow가
가능한 함수로 들어가는 패턴

Integer overflow가
가능한 함수를 통해
할당 받은 메모리가 memcpy
로 들어가는 패턴



(FreeRDP) Fuzzing with AFL ++

간단한 Generation Based-Fuzzer



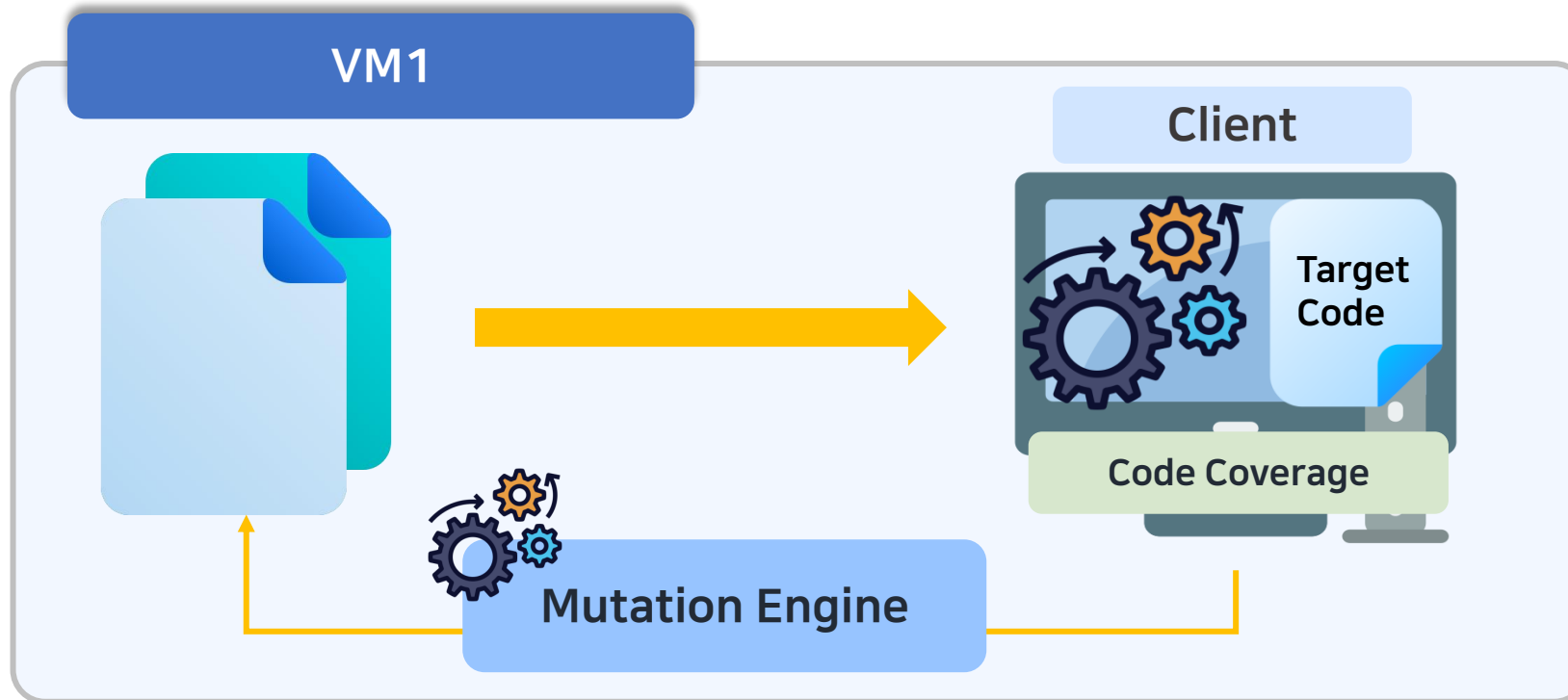
성능상 한계



AFL++
(American Fuzzy Lop plus plus)

- 1 Coverage-guided fuzzer로 확률 높이기
- 2 Compile time instrumentation (오픈소스)
- 3 Persistent 모드

(FreeRDP) Fuzzing with AFL ++



(FreeRDP) Fuzzing with AFL ++

타겟 코드

```
static UINT echo_on_data_received(IWTSVirtualChannelCallback* pChannelCallback, wStream* data)
{
    GENERIC_CHANNEL_CALLBACK* callback = (GENERIC_CHANNEL_CALLBACK*)pChannelCallback;
    BYTE* pBuffer = Stream_Pointer(data);
    UINT32 cbSize = Stream_GetRemainingLength(data);

    /* echo back what we have received. ECHO does not have any message IDs. */
    return callback->channel->Write(callback->channel, cbSize, pBuffer, NULL);
}
```

(FreeRDP) Fuzzing with AFL ++

타겟 코드

```
static UINT echo_on_data_received(IWTSVirtualChannelCallback* pChannelCallback, wStream* data)
{
    GENERIC_CHANNEL_CALLBACK* callback = (GENERIC_CHANNEL_CALLBACK*)pChannelCallback;
    BYTE* pBuffer = Stream_Pointer(data);
    UINT32 cbSize = Stream_GetRemainingLength(data);

    /* echo back what we have received. ECHO does not have any message IDs. */
    return callback->channel->Write(callback->channel, cbSize, pBuffer, NULL);
}
```

구조체1

```
typedef struct
{
    IWTSVirtualChannelCallback iface;
    IWTSPlugin* plugin;
    IWTSVirtualChannelManager* channel_mgr;
    IWTSVirtualChannel* channel;
} GENERIC_CHANNEL_CALLBACK;
```

동적 디버깅으로 각각의 값 확인,
사용하지 않는 값은 NULL로 처리

구조체2

```
{
    BYTE* buffer;
    BYTE* pointer;
    size_t length;
    size_t capacity;

    DWORD count;
    wStreamPool* pool;
    BOOL isAllocatedStream;
    BOOL isOwner;
} wStream;
```

FreeRDP Stream 관련 API 사용

(FreeRDP) Fuzzing with AFL ++

GENERIC_CHANNEL_CALLBACK

```
typedef struct
{
    IWTSVirtualChannelCallback iface;
    IWTSPlugin* plugin;
    IWTSVirtualChannelManager* channel_mgr;
    IWTSVirtualChannel* channel;
} GENERIC_CHANNEL_CALLBACK;
```

```
pwndbg> p *(GENERIC_CHANNEL_CALLBACK*)0x7ffd5d657b0
$1 = {
  iface = {
    OnDataReceived = 0x7ffff7d9bd90 <echo_on_data_received>,
    OnOpen = 0x0,
    OnClose = 0x7ffff7d9bdf0 <echo_on_close>
  }
  plugin = 0x7ffffe8fe9ee0,
  channel_mgr = 0x7ffffe800b7c0,
  channel = 0x7ffd776ce40
}
```

```
IWTSVirtualChannel channel = (IWTSVirtualChannel*)calloc(1,
sizeof(IWTSVirtualChannel));
channel
```

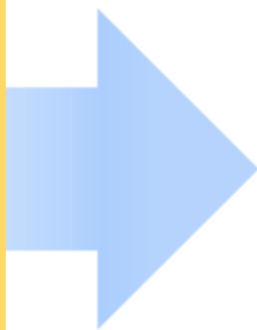
```
GENERIC_CHANNEL_CALLBACK* callback =
(GENERIC_CHANNEL_CALLBACK*)calloc(1, sizeof(GENERIC_CHANNEL_CALLBACK));
callback->iface.OnDataReceived = echo_on_data_received;
callback->iface.OnOpen = NULL;
callback->iface.OnClose = echo_on_close;
callback->plugin = NULL; /* never used */
callback->channel_mgr = NULL; /* never used */
callback->channel = NULL; /* never used */
```


(FreeRDP) Fuzzing with AFL ++

Data (wStream *)

```
typedef struct
{
    BYTE* buffer;
    BYTE* pointer;
    size_t length;
    size_t capacity;

    DWORD count;
    wStreamPool* pool;
    BOOL isAllocatedStream;
    BOOL isOwner;
} wStream;
```



```
0x7fffe52dac60, wStream* s = Stream_New(NULL, len);
if (!Stream_EnsureRemainingCapacity(s, len))
{
    /* ERROR */
    continue;
}

Stream_Write(s, buf, len);

if (Stream_Capacity(s) != Stream_GetPosition(s))
{
    /* ERROR */
    continue;
}

Stream_SealLength(s);
Stream_SetPosition(s, 0);
```

(FreeRDP) Fuzzing with AFL ++

Harness

```

while (__AFL_LOOP(UINT_MAX)) {
    len = AFL_FUZZ_TESTCASE_LEN;
    memset(callback, 0, sizeof(GENERIC_CHANNEL_CALLBACK));
    callback->iface.OnDataReceived = echo_on_data_received;
    callback->iface.OnOpen = NULL;
    callback->iface.OnClose = echo_on_close;
    callback->plugin = NULL;
    callback->channel_mgr = NULL;
    callback->channel = NULL;

    wStream* s = Stream_New(NULL, len);
    if (!Stream_EnsureRemainingCapacity(s, len))
    {
        continue;
    }

    Stream_Write(s, buf, len);

    if (Stream_Capacity(s) != Stream_GetPosition(s))
    {
        continue;
    }

    Stream_SealLength(s);
    Stream_SetPosition(s, 0);

    /* target function */
    echo_on_data_received(callback, s);
}

```

구조체1

구조체2

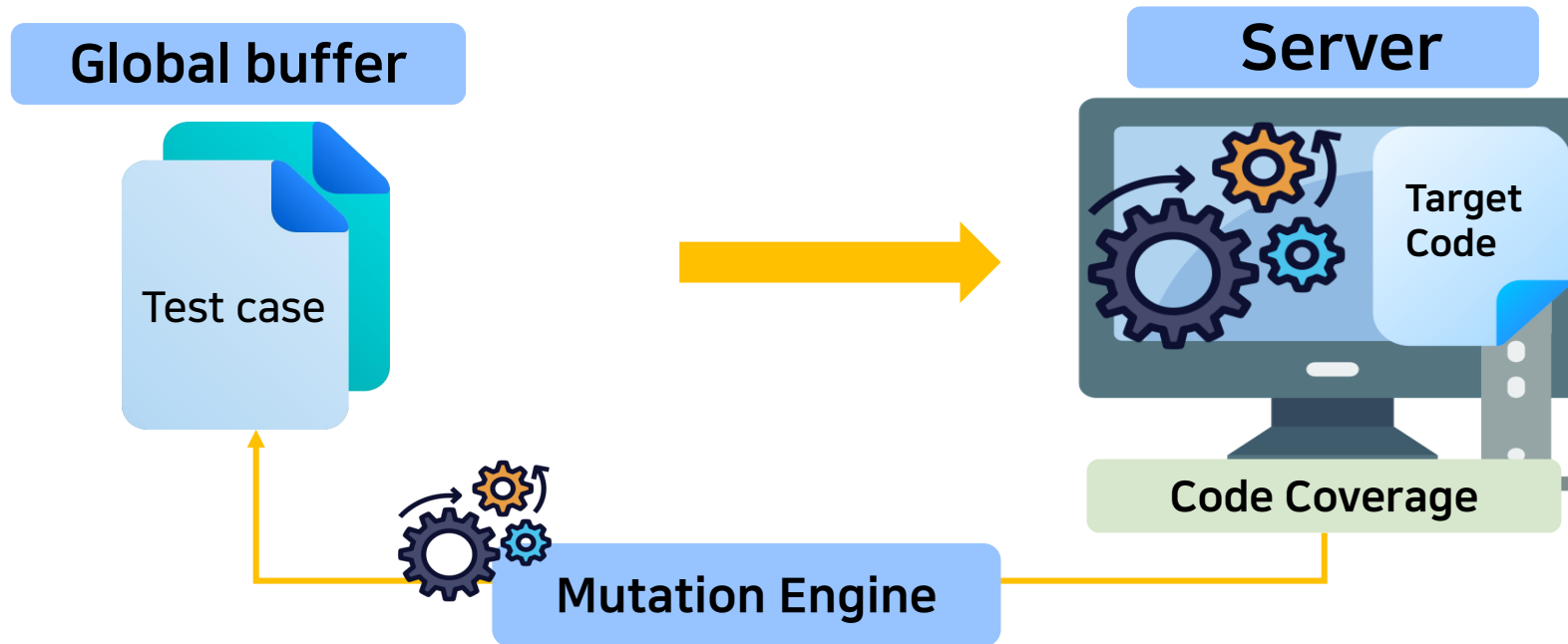
타겟 함수명

```

american fuzzy lop ++4.04a {default} (./fuzzme) [fast]
- process timing -
  run time : 0 days, 0 hrs, 0 min, 23 sec
  last new find : none yet (odd, check syntax!)
  last saved crash : none seen yet
  last saved hang : none seen yet
- cycle progress -
  now processing : 0.1723 (0.0%)
  runs timed out : 0 (0.00%)
- stage progress -
  now trying : havoc
  stage execs : 1174/1175 (99.91%)
  total execs : 2.02M
  exec speed : 88.2k/sec
- fuzzing strategy yields -
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 0/2.02M, 0/0
  py/custom/rq : unused, unused, unused, unused
  trim/eff : 33.33%/1, disabled
- overall results -
  cycles done : 574
  corpus count : 1
  saved crashes : 0
  saved hangs : 0
- map coverage -
  map density : 12.31% / 12.31%
  count coverage : 61.00 bits/tuple
- findings in depth -
  favored items : 1 (100.00%)
  new edges on : 1 (100.00%)
  total crashes : 0 (0 saved)
  total tmouts : 3 (0 saved)
- item geometry -
  levels : 1
  pending : 0
  pend fav : 0
  own finds : 0
  imported : 0
  stability : 100.00%
[cpu001: 37%]

```

(xrdp) Fuzzing with AFL ++



(MSRDP) Fuzzing with Jackalope

```
Total execs: 1798279198  
Unique samples: 4 (0 discarded)  
Crashes: 0 (0 unique)  
Hangs: 1  
Offsets: 2155  
Execs/s: 1
```



Jackalope

Windows & MacOS

Blackbox Binary 대상,
Coverage guided fuzzer

WinAFL을 만든 Project Zero에서
제작한 Fuzzer

선택 이유

가벼운
Instrumentation

관심있는 모듈인
mstscax.dll에 대해서만
Instrumentation

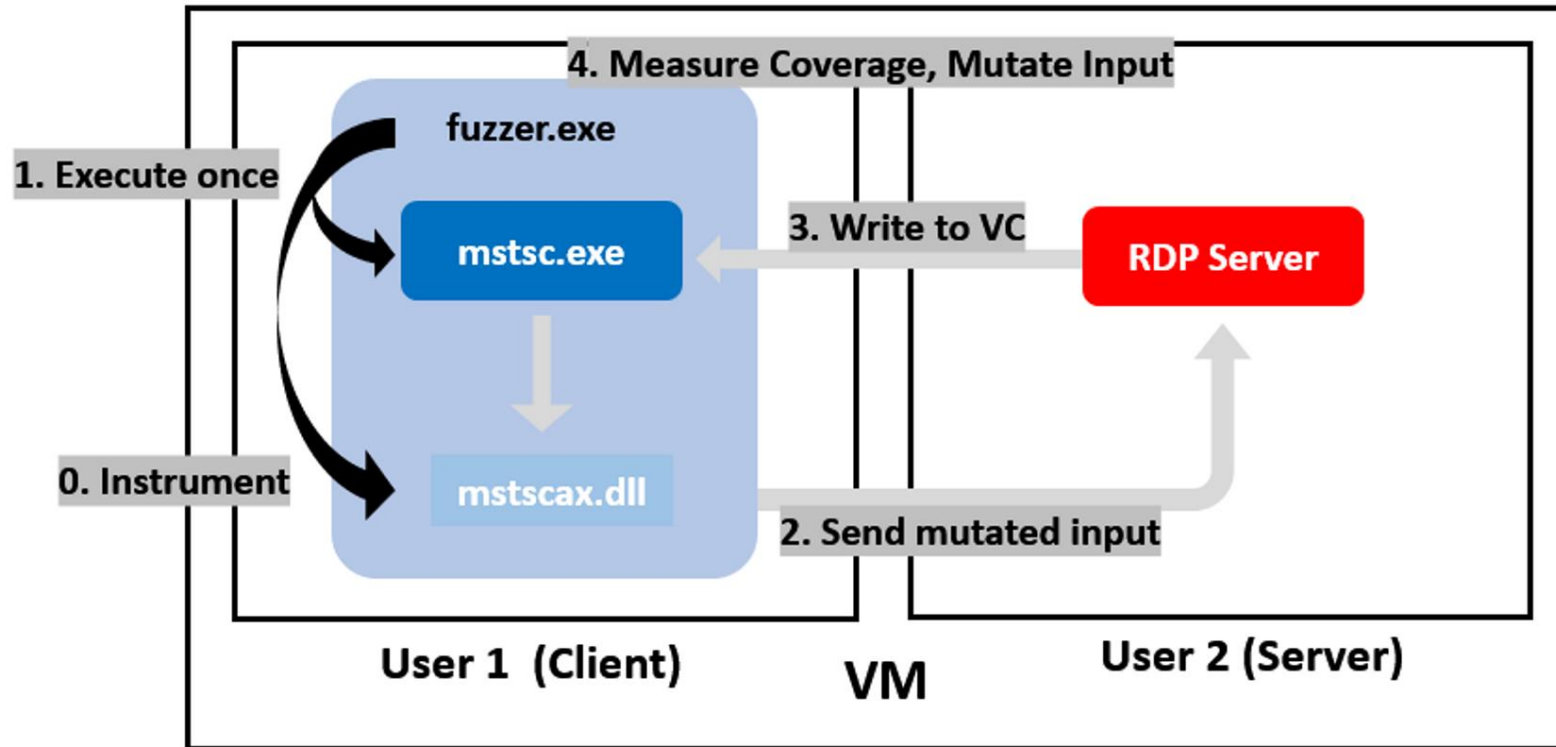
Grammar
Based

MS RDP는 PDU 포맷에 맞지
않는 데이터가 들어올 경우
무시하거나 연결 끊음

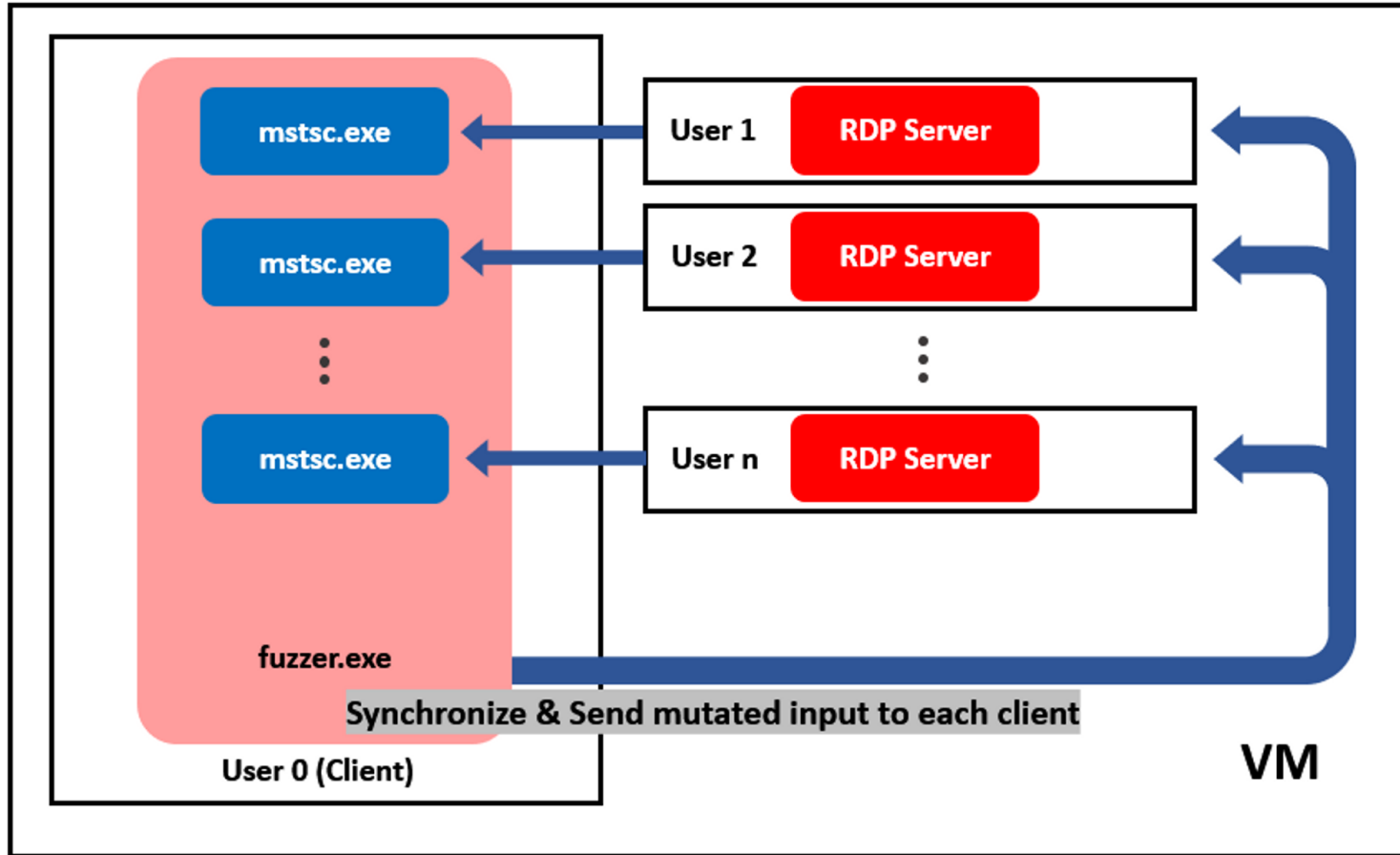
용이한
Custom

Mutator, Sample Delivery
용이하게 변경 가능함

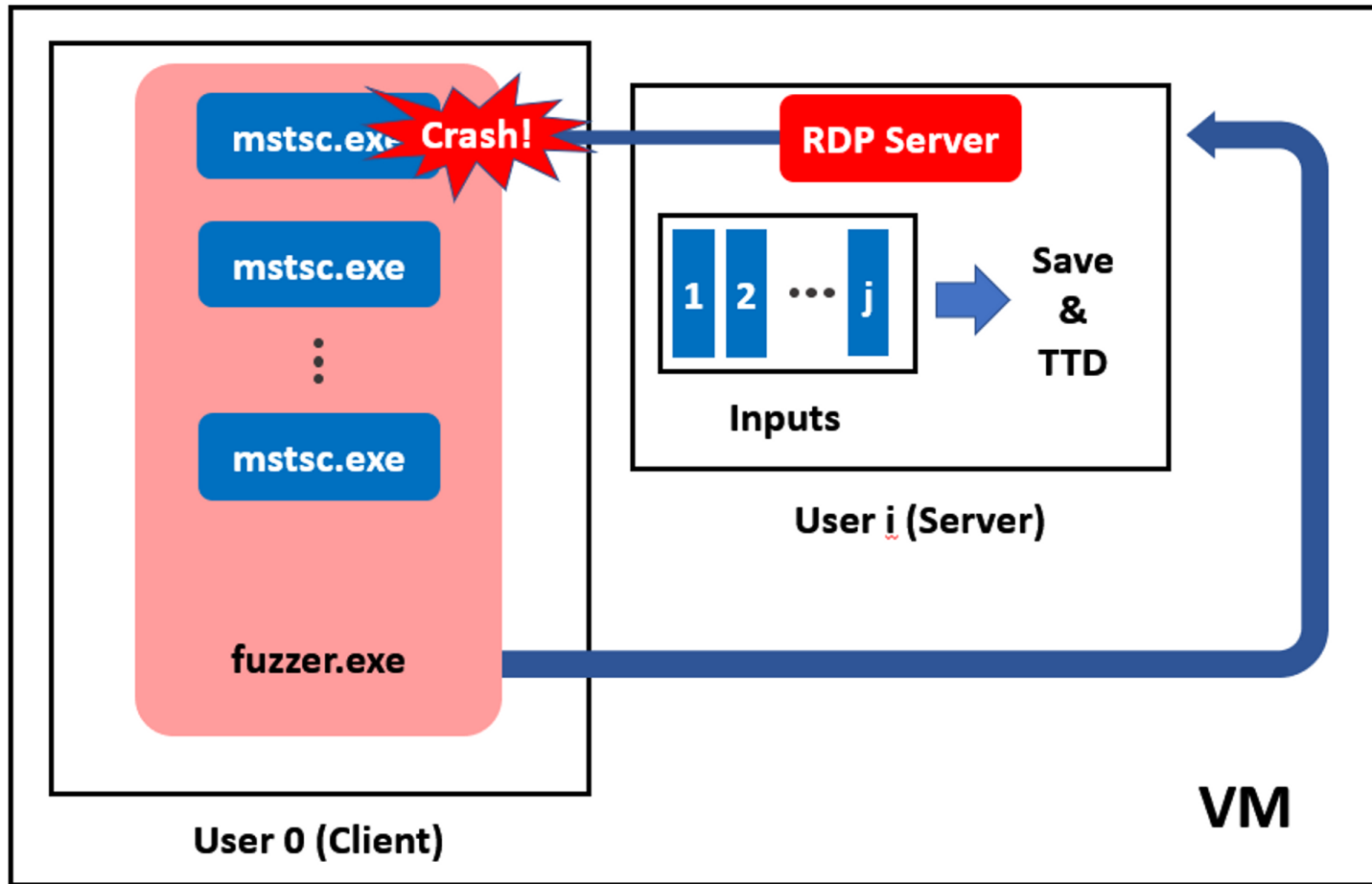
(MSRDP) Fuzzing with Jackalope



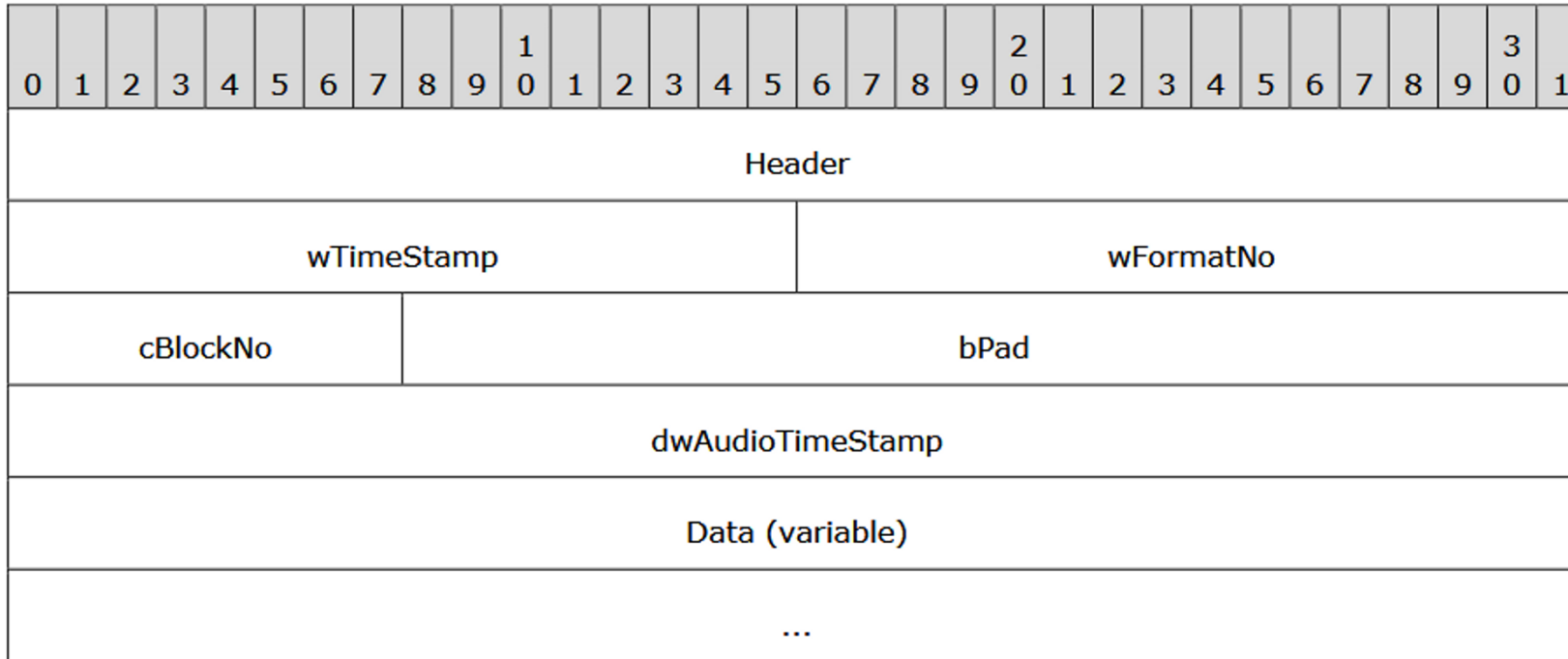
(MSRDP) Fuzzing with Jackalope



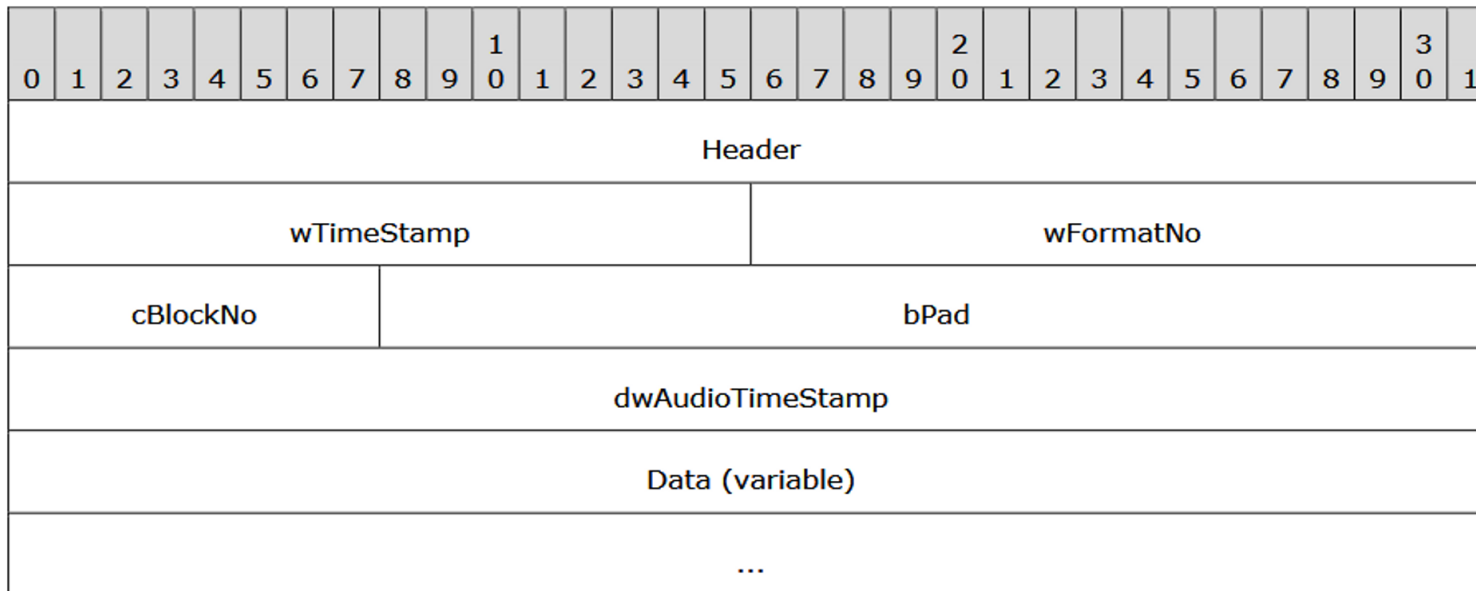
(MSRDP) Fuzzing with Jackalope



(MSRDP) Fuzzing with Jackalope



(MSRDP) Fuzzing with Jackalope



Grammar

Jackalope: AST 기반 Grammar

➔ 더 복잡한 문법구조의 RDP PDU

ex.
특정 필드 길이의 PDU 메타데이터 필드

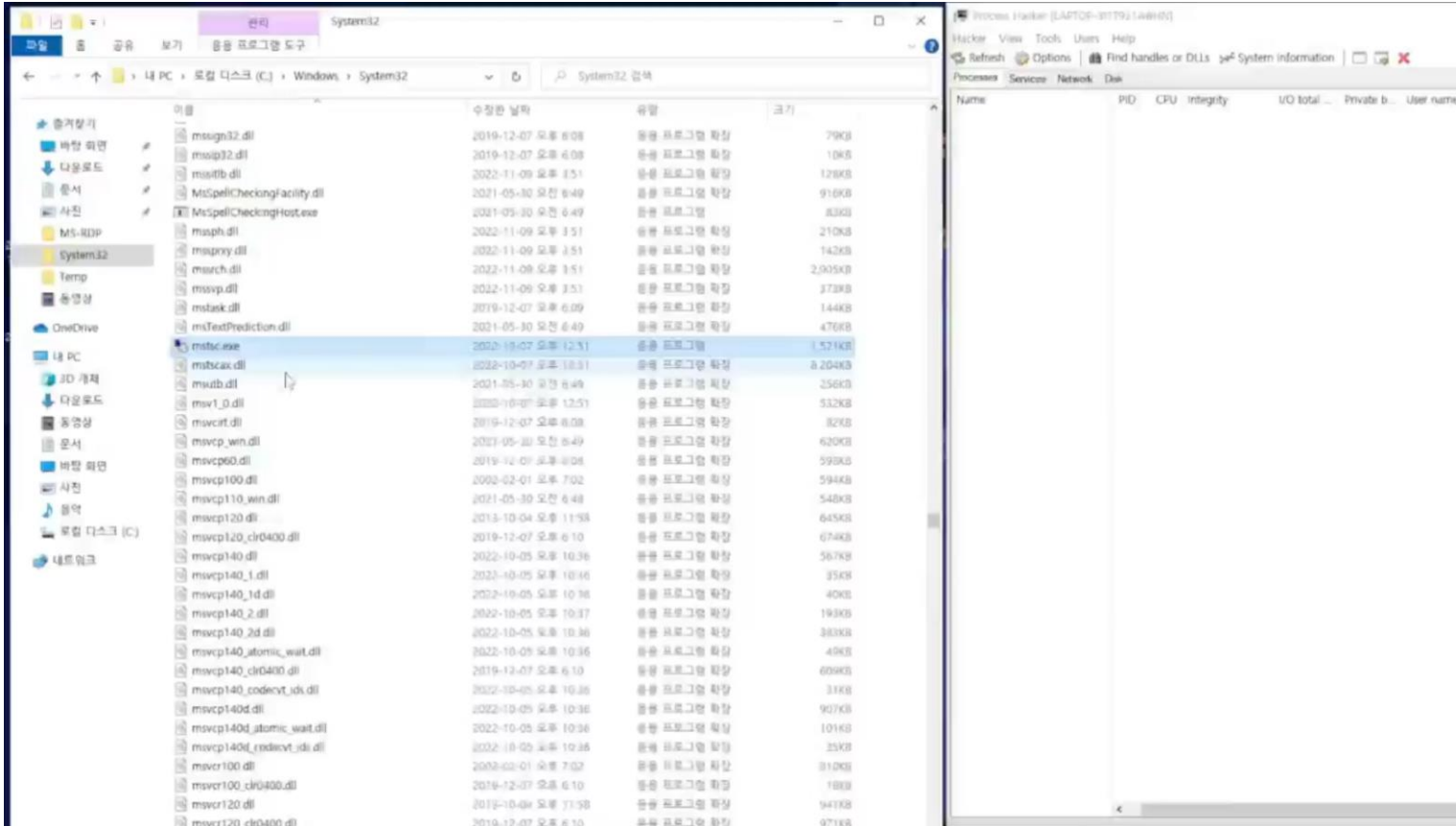
```
<root>=<0x0dff><length size=2 exclude><word><word><byte><0xffffffff><dword><repeat_bytes min=0 max=65535 p=0.99>
```

(MSRDP) Fuzzing with Jackalope

```
선택 C:\Windows\System32\cmd.exe - fuzzer.exe -in - -resume -out out_8 -t 1000 -delivery shmem -instrument_module mfplat.dll -target_mod...  
Offsets: 2155  
Execs/s: 4725  
Fuzzing sample 00001  
Instrumented module mfplat.dll, code size: 1503232  
Fuzzing sample 00000  
Fuzzing sample 00002  
  
Total execs: 1798279196  
Unique samples: 4 (0 discarded)  
Crashes: 0 (0 unique)  
Hangs: 1  
Offsets: 2155  
Execs/s: 3196  
Exception at address 00007FFD533EE625  
Access address: 000001DA03994FF0  
  
Total execs: 1798279197  
Unique samples: 4 (0 discarded)  
Crashes: 0 (0 unique)  
Hangs: 1  
Offsets: 2155  
Execs/s: 1  
  
Total execs: 1798279198  
Unique samples: 4 (0 discarded)  
Crashes: 0 (0 unique)  
Hangs: 1  
Offsets: 2155  
Execs/s: 1
```

결론

(MSRDP) CVE-2023-28267



➔ PoC를 통해 구한 클라이언트의 이미지 베이스 주소 = 실제 로드된 모듈 주소

What did we find?

| Target | CVE | Vulnerability |
|---------|----------------|---|
| FreeRDP | CVE-2022-39282 | Read of uninitialized memory with parallel port redirection |
| | CVE-2022-39283 | RDP client read out of bounds data and display it |
| | CVE-2022-39316 | Out of bound read in zgfx decoder |
| | CVE-2022-39317 | Undefined behaviour in zgfx decoder |
| | CVE-2022-39318 | Division by zero in urbdrc channel |
| | CVE-2022-39319 | Missing length validation in urbdrc channel |
| | CVE-2022-39320 | Heap buffer overflow in urbdrc channel |
| | CVE-2022-39347 | Missing input length validation in `drive` channel |
| | CVE-2022-41877 | Missing path sanitation with `drive` channel |

What did we find?

| Target | CVE | Vulnerability |
|--|----------------|---|
| XRDP | CVE-2022-23468 | Buffer Overflow in xrdp_login_wnd_create |
| | CVE-2022-23477 | Buffer Overflow in audin_send_open |
| | CVE-2022-23478 | Out of Bound Write in xrdp_mm_trans_process_drdynvc_channel_open |
| | CVE-2022-23479 | Buffer Overflow in xrdp_mm_chan_data_in |
| | CVE-2022-23480 | Buffer Overflow in devredir_proc_client_devlist_announce |
| | CVE-2022-23481 | Out of Bound Read in xrdp_caps_process_confirm_active |
| | CVE-2022-23482 | Out of Bound Read in xrdp_caps_process_confirm_active |
| | CVE-2022-23483 | Out of Bound Read in libxrdp_send_to_channel |
| | CVE-2022-23484 | Integer Overflow in xrdp_mm_process_rail_update_window |
| | CVE-2022-23493 | Out of Bound Read in xrdp_mm_trans_process_drdynvc_channel_close |
| Microsoft Terminal Service Client | CVE-2023-28267 | Microsoft Windows Remote Desktop Connection Uninitialized Variable Information Disclosure Vulnerability |

How did we contribute?

2.9.0

Noteworthy changes:

- Backported [#8252](#): Support sending server redirection PDU
- Backported [#8406](#): Ensure X11 client cursor is never smaller 1x1
- Backported [#8403](#): Fixed multiple client side input validation issues (CVE-2022-39316, CVE-2022-39317, CVE-2022-39318, CVE-2022-39319, CVE-2022-39320, CVE-2022-41877, CVE-2022-39347)
- Backported [#7282](#): Proxy server now discards input events sent before activation was received
- Backported [#8324](#): Internal replacements for md4, md5 and hmac-md5
For the time being the RDP protocol requires these outdated hash algorithms. So any distribution that wants to ship a working FreeRDP should check the options WITH_INTERNAL_MD4 (and depending on OpenSSL deprecation status WITH_INTERNAL_MD5)

Fixed issues:

- Backported [#8341](#): Null checks in winpr_Digest_Free
- Backported [#8335](#): Missing NULL return in winpr_Digest_New
- Backported [#8192](#): Support for audin version 2 microphone channel
- Backported [#7282](#): Discard input events before activation (Fixes [#8374](#))

For a complete and detailed change log since the last release run:
git log 2.8.1..2.9.0

Thanks to "Team BT5 (BoB 11th)" for reporting the security issues.

xrdp v0.9.21

Release notes for xrdp v0.9.21 (2022/12/10)

General announcements

- Running xrdp and xrdp-sesman on separate hosts is still supported by this release, but is now deprecated. This is not secure. A future v1.0 release will replace the TCP socket used between these processes with a Unix Domain Socket, and then cross-host running will not be possible.

Security fixes

This update is recommended for all xrdp users and provides following important security fixes:

- [CVE-2022-23468](#)
- [CVE-2022-23477](#)
- [CVE-2022-23478](#)
- [CVE-2022-23479](#)
- [CVE-2022-23480](#)
- [CVE-2022-23481](#)
- [CVE-2022-23483](#)
- [CVE-2022-23482](#)
- [CVE-2022-23484](#)
- [CVE-2022-23493](#)

These security issues are reported by [Team BT5 \(BoB 11th\)](#). We appreciate their great help with making and reviewing patches.

More about RDP security..

Gitbook

The screenshot shows a Gitbook page for 'RDP Security Research'. The left sidebar contains a table of contents with the following items:

- Introduction
- Introducing Team Members
- Before Getting Started
 - What Is Remote Desktop Protocol?
 - Related Research
- Fuzzing RDP
 - AFL++ Based Fuzzing
 - Jackalope Based Fuzzing** (highlighted)
- Code Auditing RDP
 - Using CodeQL
- Exploitation Series
 - FreeRDP RCE Exploit & Write-up
 - xrdp Privilege Escalation Exploit & Write-up

The main content area is titled 'Fuzzing' and contains the following text:

Fuzzing

To start actual fuzzing with Jackalope, we referred to Black Hat talk in 2019, Europe which was about fuzzing RDP with WinAFL. We ran fuzzing agent in RDP sever side, which receives fuzzing input from RDP client and sends it to client with WTS API. To better reproduce the crash, we also saved all fuzzing inputs until the crash occurred, and replayed it with time-travel debugging.

Environment Setting (Windows)

Test environment: Windows 10 21H2 (OS build 19044.1288)

Local RDP Server

We used RDP Wrapper to construct local RDP server. With RDP Wrapper, we can connect to other user in same machine through RDP, using loopback IP address(for example, 127.0.0.2). It simplifies and efficienates our fuzzing structure.

First, download RDP Wrapper and update files.

```
wget -Uri https://github.com/stascorp/rdpwrap/releases/download/v1.6.2/RDPWrap-v1.
wget -Uri https://github.com/asmtron/rdpwrap/raw/master/autoupdate.zip -Ooutfile au
```

<https://bob11-btss-organization.gitbook.io/rdp-security-research/>

Thank you
