black hat ASIA 2025
APRIL 3-4, 2025
BRIEFINGS

One Bug to Rule Them All: Stably Exploiting a Preauth RCE Vulnerability on Windows Server 2025

Speakers:

Dr. Zhiniang Peng @ HUST & Cyber-Kunlun
Zishan Lin
Ver

WhoAmI

Zhiniang Peng @edwardzpeng
Associate Professor @HUST
Part-time Researcher @Cyber-Kunlun

PhD in Cryptography, Published many research in both Industry & Academia

More about me:https://sites.google.com/site/zhiniangpeng

HUST: Huazhong University of Science and Technology

Cyber-Kunlun: World-Leading Security Research Lab in China

WhoAmI

Zishan Lin

Security Researcher

Focus on Windows Security for years

WhoAmI

Ver @Ver0759

Security Researcher

Focus on Windows/IOT/BlockChain Security for three years

MSRC MVR

Agenda

- Introduction
- Remote Desktop License RPC Service Internals
- CVE-2024-38077
- Exploitation Overview
- Playing with the LFH
- Summary

Introduction

Preauth bugs & exploitations

- The history of Windows Preauth RCE Memory Corruption Exploitation

  CVE-2014-6321 WinShock

  CVE-2017-0144 Eternal Blue

  CVE-2019-0708 Bluekeep

  CVE-2020-0796 SMBGhost

  ……

- Every year there are **numerous** preauth memory corruption vulnerabilities in Windows, but **only a few of** these are really have exploitation.

1	CVE-2024-21334	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
2	CVE-2024-21407	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
3	CVE-2024-21416	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
4	CVE-2024-30020	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
5	CVE-2024-30080	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
6	CVE-2024-35264	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
7	CVE-2024-38045	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
8	CVE-2024-38053	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
9	CVE-2024-38063	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-191: Integer Underflow (Wrap or Wraparound)
10	CVE-2024-38074	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-191: Integer Underflow (Wrap or Wraparound)
11	CVE-2024-38076	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
12	CVE-2024-38077	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
13	CVE-2024-38078	CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
14	CVE-2024-38119	CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
15	CVE-2024-38140	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
16	CVE-2024-38199	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
17	CVE-2024-38229	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
18	CVE-2024-43447	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-415: Double Free
19	CVE-2024-43491	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RL:O/RC:C	CWE-416: Use After Free
20	CVE-2024-43566	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N/E:U/RL:O/RC:C	CWE-190: Integer Overflow or Wraparound
21	CVE-2024-43582	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
22	CVE-2024-43587	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
23	CVE-2024-43598	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-122: Heap-based Buffer Overflow
24	CVE-2024-49106	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
25	CVE-2024-49108	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
26	CVE-2024-49112	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-190: Integer Overflow or Wraparound
27	CVE-2024-49115	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
28	CVE-2024-49116	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
29	CVE-2024-49118	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
30	CVE-2024-49122	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
31	CVE-2024-49126	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
32	CVE-2024-49127	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
33	CVE-2024-49128	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free
34	CVE-2024-49132	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C	CWE-416: Use After Free

Part of memory corruption vulnerabilities in 2024

Mitigation in Windows

- Mitigation in modern Windows
  - Control Flow Guard (CFG)
  - GuardStack (GS) / Canary
  - Arbitrary Code Guard (ACG)
  - Control-Flow Enforcement Technology (CET)
  - Address Space Layout Randomization (ASLR)
  - ……

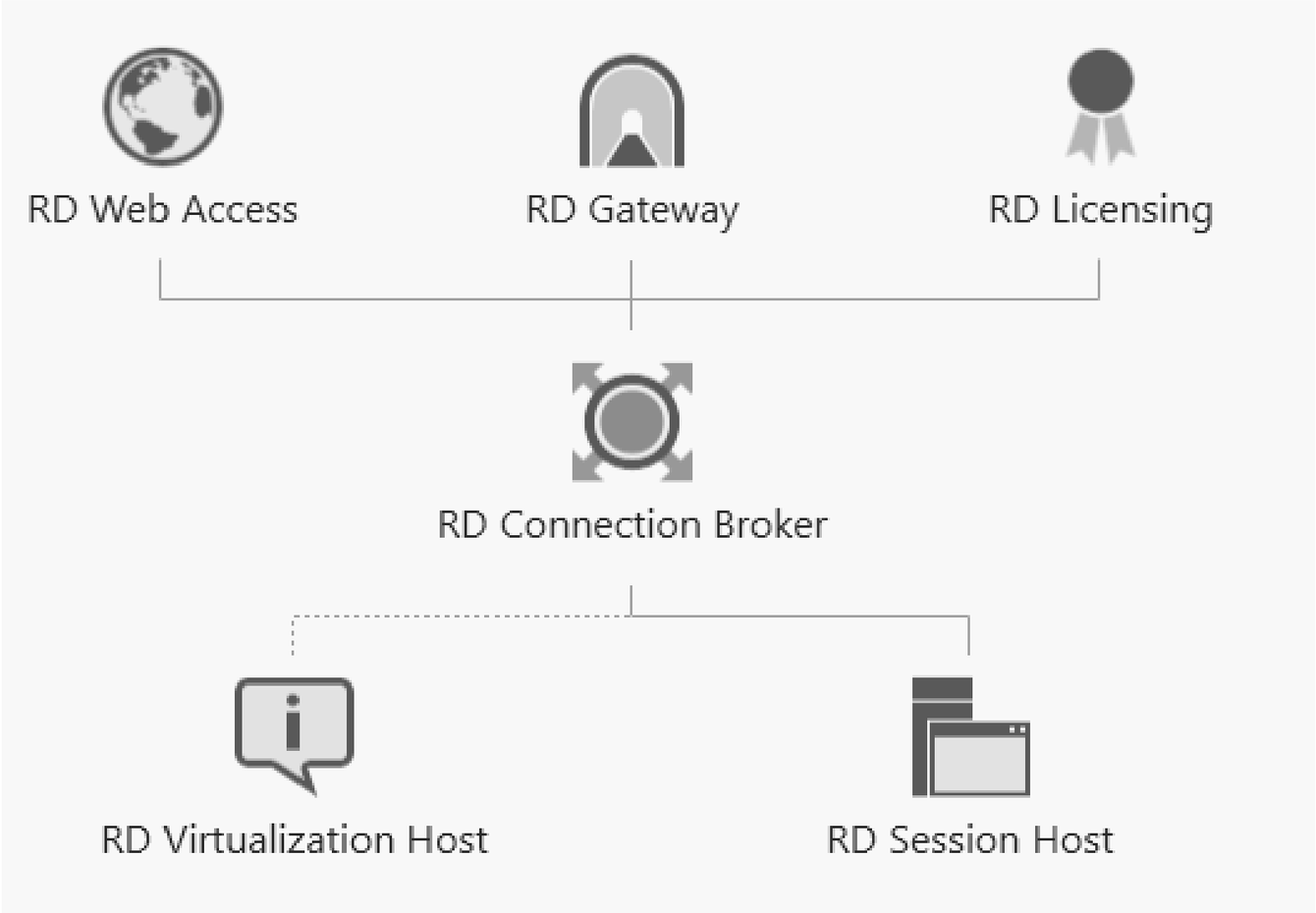
- CVE-2024-38077 Madlicense
  - Windows Remote Desktop License Server Preauth RCE
  - One bug to Rule Them All

Remote Desktop Service

- Remote Desktop Service is a core service on Windows
  - Personal Remote Access
  - Server Management
  - Remote Desktop Rental Services

- Many historical security vulnerabilities
  - From CVE-2012-0002 to CVE-2024-43599
  - No remote exploit for many years

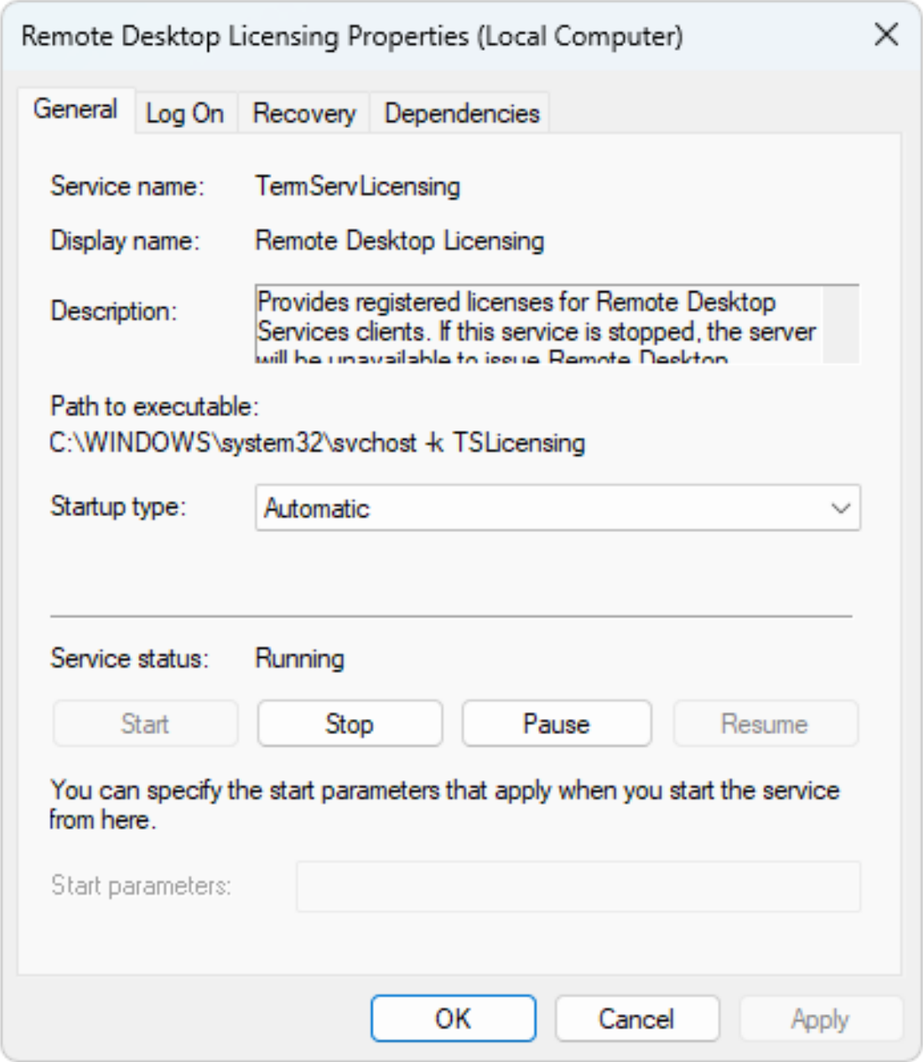
Remote Desktop Service Framework



Remote Desktop License Service

- Client will need a Client Access License (CAL) to access RD Session.

- 170, 000+ RDL exposed to the Internet

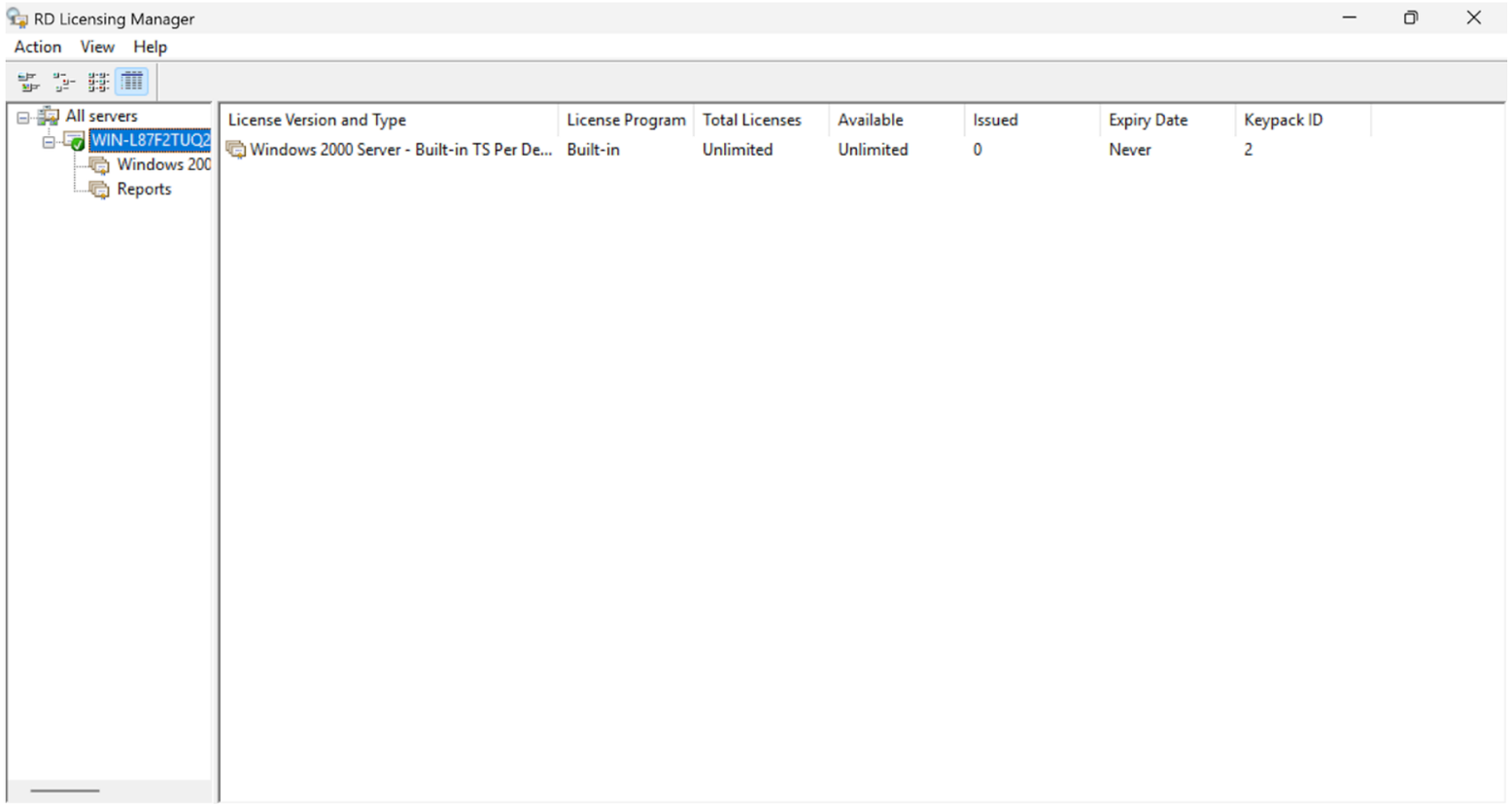


black hat ASIA 2025
APRIL 3-4, 2025
BRIEFINGS

Remote Desktop License Service RPC Internals

Remote Desktop License Service Internals

- Using RD Licensing Manager to manager RD Licensing Server



Remote Desktop License Service RPC

- Provided RPC interface
- Lack of authentication(Flags == 0x0 and no Security Callback)
- 119 RPC interfaces in lserver.dll

Interfaces

Pid	Uuid	Ver	Type	Procs	Flags	Callback	Stub
3120	18f70770-8e64-11cf-9af1-0020af6e72f4	0.0	RPC	5	0x21	0x00007ff81c9f5...	Interpreted
3120	3d267954-eeb7-11d1-b94e-00c04fa3080d	1.0	RPC	119	0x0		Interpreted

Procedures

Index	Name	Address	Format
0	TLSRpcGetVersion	0x00007ff80c1bf1f0	0x00007ff80c258fb2
1	TLSRpcConnect	0x00007ff80c1bc710	0x00007ff80c258fe4
2	TLSRpcDisconnect	0x00007ff80c1bcb10	0x00007ff80c25900e
3	TLSRpcSendServerCertificate	0x00007ff80c1c3f20	0x00007ff80c25903a
4	TLSRpcGetServerName	0x00007ff80c1be4a0	0x00007ff80c259078
5	TLSRpcGetServerScope	0x00007ff80c1be9f0	0x00007ff80c2590b6
6	TLSRpcGetInfo	0x00007ff80c1bdd70	0x00007ff80c2590f4
7	TLSRpcIssuePlatformChallenge	0x00007ff80c1bfbf0	0x00007ff80c25914a
8	TLSRpcRequestNewLicense	0x00007ff80c1c2630	0x00007ff80c259194
9	TLSRpcUpgradeLicense	0x00007ff80c1c4db0	0x00007ff80c2591fc
10	TLSRpcAllocateConcurrentLicense	0x00007ff80c1bb7f0	0x00007ff80c25925e
11	TLSRpcGetLastError	0x00007ff80c1bdd80	0x00007ff80c2592a2
12	TLSRpcKeyPackEnumBegin	0x00007ff80c1c0120	0x00007ff80c2592e0
13	TLSRpcKeyPackEnumNext	0x00007ff80c1c03f0	0x00007ff80c259324

RPC-Based Calls

- TLSRpcConnect

  Malloc context struct and return the handle

- context->clientFlags

  None Access -> 0x00000000

  Normal Access -> 0x00000001, 0x00000004, 0x00000008

  Admin Access -> 0x00000002

  Full Access -> 0xFFFFFFFF

- Normal Access: Limited access

  TLSRpcGetVersion

  TLSRpcGetSupportFlags

  TLSRpcSendServerCertificate

  TLSRpcGetServerName

  ……

- If set context->clientFlags = 0xFFFFFFFF

  We can call all RPC methods in lserver.dll!

```
TLSRpcConnect(
    [in] handle_t binding,
    [out] PCONTEXT_HANDLE *pphContext
);
```

```
typedef struct context {
    LPTSTR   clientName;
    DWORD    refCount;
    DWORD    clientFlags;
    DWORD    lastError;
    DWORD    contextType;
    HANDLE   contextHandle;
}
```

TLSRpcChallengeServer and TLSRpcResponseServerChallenge

- TLSRpcChallengeServer

  Get pServerChallenge

- TLSRpcResponseServerChallenge

  Send pClientResponse

  Set clientFlags to 0xFFFFFFFF

- How to calculate pClientResponse

  md5(pServerChallenge + fixed_Guid)

- Congratulations! You beat the Microsoft homemade cryptosystem and make your context have full access to all the RPC method.

```
TLSRpcChallengeServer(
    [in] PCONTEXT_HANDLE phContext,
    [in] DWORD dwClientType,
    [in] PTLSCHALLENGEDATA pClientChallenge,
    [out][in] PTLSCHALLENGERESPONSEDATA* pServerResponse,
    [out][in] PTLSCHALLENGEDATA* pServerChallenge,
    [ref][out][in] PDWORD pdwErrCode
);
```

```
TLSRpcResponseServerChallenge(
    [in] PCONTEXT_HANDLE phContext,
    [in, ref] PTLSCHALLENGERESPONSEDATA pClientResponse,
    [in, out, ref] PDWORD pdwErrCode
);
```

CVE-2024-38077

TLSRpcTelephoneRegisterLKP

- TLSRpcTelephoneRegisterLKP

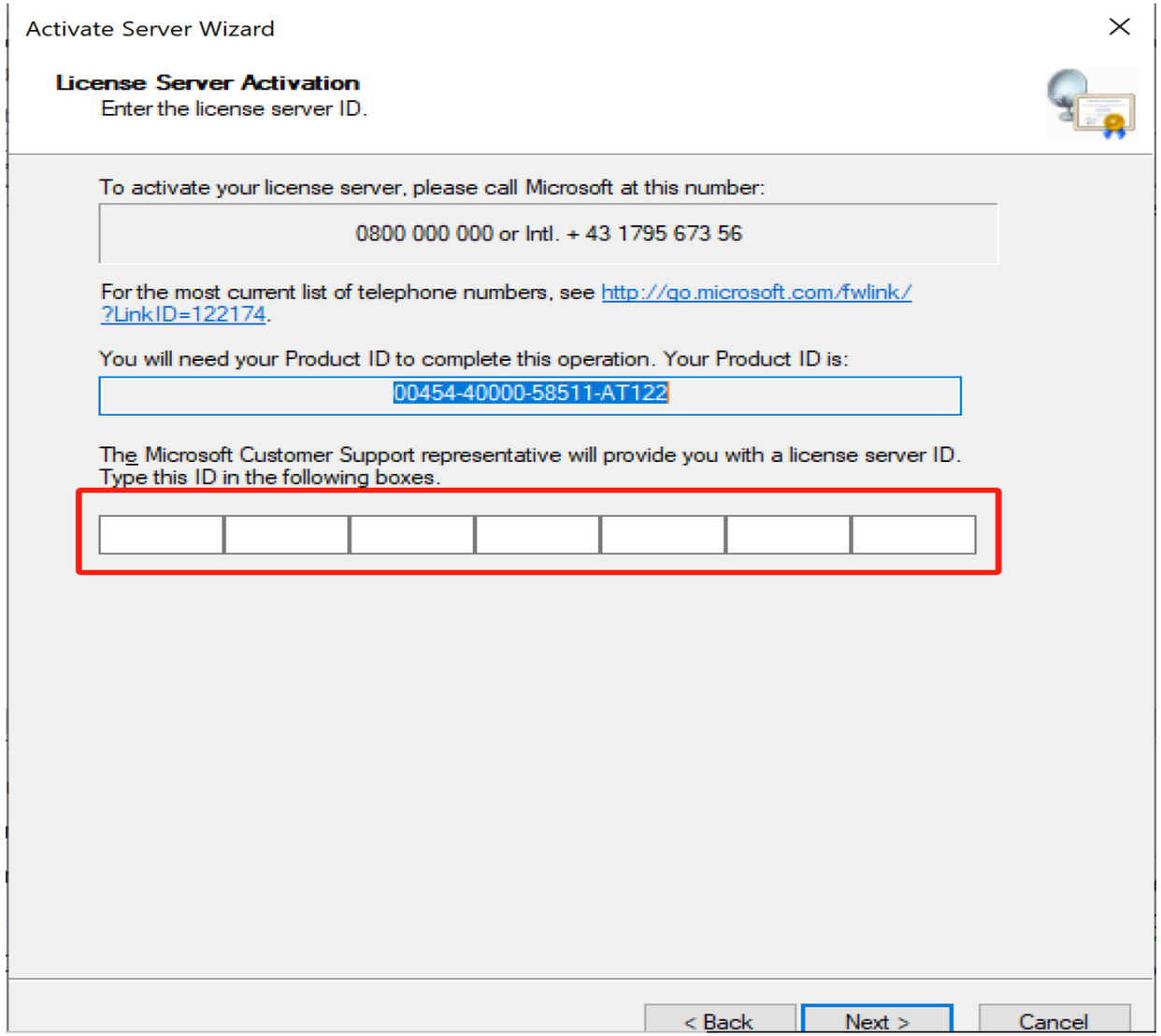
  Telephone activation

- License Server ID

  License Key Pack (LKP)

  Using Product ID to get it

```
TLSRpcTelephoneRegisterLKP(
    [in] PCONTEXT_HANDLE phContext,
    [in] DWORD cbData,
    [in, size_is(dwData)] PBYTE pbData,
    [out, ref] PDWORD pdwErrCode
);
```



CVE-2024-38077

- Heap overflow in CDataCoding::DecodeData

  TLSRpcTelephoneRegisterLKP -> TLSDBTelephoneRegisterLicenseKeyPack -> LKPLiteVerifyLKP -> B24DecodeMSID -> CDataCoding::DecodeData

- LKPLiteVerifyLKP

  Base24 Decoding using input

  Fixed-size heap (0x20)

- Unrestricted heap overflow

  Controllable contents

  Low Fragmentation Heap (LFH)

```
(c30.144c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
ntdll!RtlpHpTagFreeHeap+0x5cc:
00007ff8`1e2d78dc 4c8b44d1f8      mov     r8,qword ptr [rcx+rdx*8-8] ds:000001ee`06392ff0=????????????????
0:011> k
 # Child-SP          RetAddr           Call Site
00 000000e5`1737b600 00007ff8`1e2d6915 ntdll!RtlpHpTagFreeHeap+0x5cc
01 000000e5`1737b690 00007ff8`0c207062 ntdll!RtlFreeHeap+0x285
02 000000e5`1737b780 00007ff8`0c1ec793 lserver!LKPLiteVerifyLKP+0xae
03 000000e5`1737b7d0 00007ff8`0c1c43eb lserver!TLSDBTelephoneRegisterLicenseKeyPack+0x163
04 000000e5`1737d1e0 00007ff8`1d3f1913 lserver!TLSRpcTelephoneRegisterLKP+0x15b
05 000000e5`1737ead0 00007ff8`1d3374ed RPCRT4!Invoke+0x73
06 000000e5`1737eb30 00007ff8`1d336f6a RPCRT4!NdrStubCall2+0x30d
07 000000e5`1737ede0 00007ff8`1d3a3897 RPCRT4!NdrServerCall2+0x1a
08 000000e5`1737ee10 00007ff8`1d3560f4 RPCRT4!DispatchToStubInCNoAvrf+0x17
09 000000e5`1737ee60 00007ff8`1d3552b4 RPCRT4!RPC_INTERFACE::DispatchToStubWorker+0x194
0a 000000e5`1737ef30 00007ff8`1d36b61a RPCRT4!RPC_INTERFACE::DispatchToStub+0x1f4
0b 000000e5`1737f1d0 00007ff8`1d36b3c1 RPCRT4!OSF_SCALL::DispatchHelper+0x13a
0c 000000e5`1737f2f0 00007ff8`1d36a0d9 RPCRT4!OSF_SCALL::DispatchRPCCall+0x89
0d 000000e5`1737f320 00007ff8`1d368c68 RPCRT4!OSF_SCALL::ProcessReceivedPDU+0xe1
0e 000000e5`1737f3c0 00007ff8`1d3807ec RPCRT4!OSF_SCONNECTION::ProcessReceiveComplete+0x34c
0f 000000e5`1737f4c0 00007ff8`1ba158a1 RPCRT4!CO_ConnectionThreadPoolCallback+0xbc
10 000000e5`1737f540 00007ff8`1e2950a7 KERNELBASE!BasepTpIoCallback+0x51
11 000000e5`1737f590 00007ff8`1e24934e ntdll!TppIopExecuteCallback+0x1b7
12 000000e5`1737f610 00007ff8`1cc61fd7 ntdll!TppWorkerThread+0x57e
13 000000e5`1737f970 00007ff8`1e31b66c KERNEL32!BaseThreadInitThunk+0x17
14 000000e5`1737f9a0 00000000`00000000 ntdll!RtlUserThreadStart+0x2c
```

Exploitation Overview

Exploitation

- Key Points in the exploit
  - Leak Address and bypass ASLR
    - heap address, ntdll address, PEB address, PE address, dll address, …
  - Hijack Rip
    - Bypass CFG

- One bug to Rule Them All
  - CVE-2024-38077
  - Using one vulnerability to defeat all the mitigation on the latest Windows Server

Finding the victim: Context in TLSRpcConnect

- Struct context will malloc in TLSRpcConnect
  - Size = 0x20
  - Two key pointers
- clientName -> Leak address
  - TLSRpcRetrieveTermServCert
- contextHandle -> Hijack control flow
  - TLSRpcKeyPackEnumNext

```
typedef struct context {
    LPTSTR   clientName;
    DWORD    refCount;
    DWORD    clientFlags;
    DWORD    lastError;
    DWORD    contextType;
    HANDLE   contextHandle;
}
```

TLSRpcRequestTermServCert and TLSRpcRetrieveTermServCert

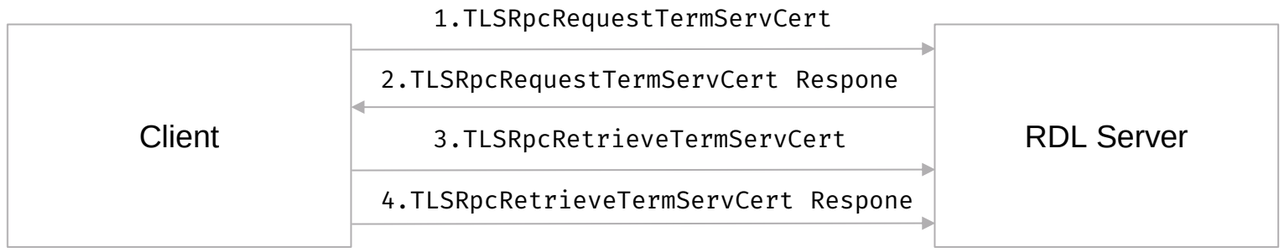
- TLSRpcRequestTermServCert

  Request the Terminal Server Cert

- TLSRpcRetrieveTermServCert

  Retrieve the Terminal Server Cert

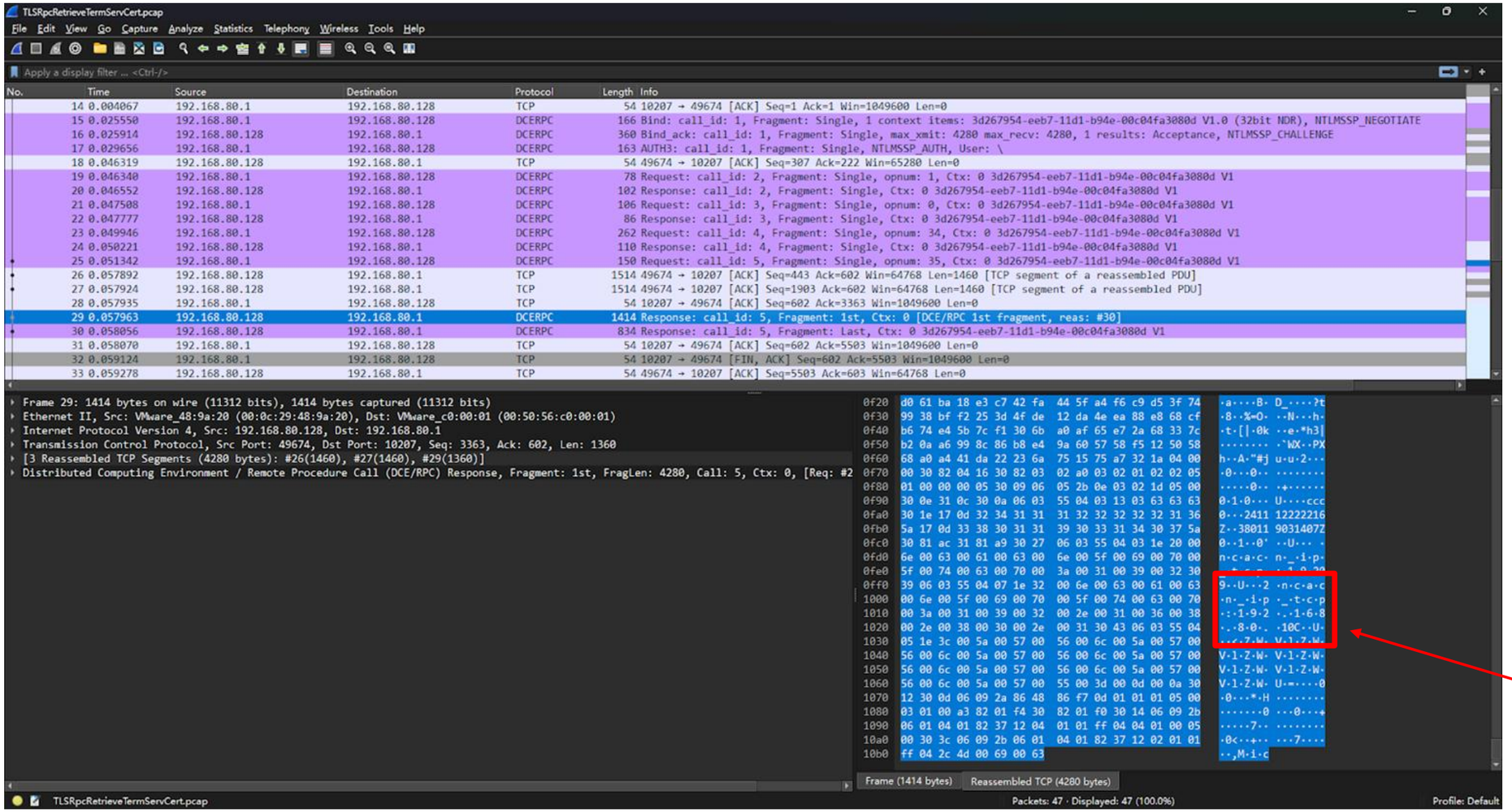
  pbCert contains context->clientName




```
TLSRpcRequestTermServCert(
    [in] PCONTEXT_HANDLE phContext,
    [in] PTLSHYDRACERTREQUEST pbRequest,
    [in, out, ref] PDWORD cbChallengeData,
    [out, size_is(, *cbChallengeData)] PBYTE*
pbChallengeData,
    [in, out, ref] PDWORD pdwErrCode
);
```

```
TLSRpcRetrieveTermServCert(
    [in] PCONTEXT_HANDLE phContext,
    [in] DWORD cbResponseData,
    [in, size_is(cbResponseData)] PBYTE
pbResponseData,
    [in, out, ref] PDWORD cbCert,
    [out, size_is(, *cbCert)] PBYTE* pbCert,
    [in, out, ref] PDWORD pdwErrCode
);
```

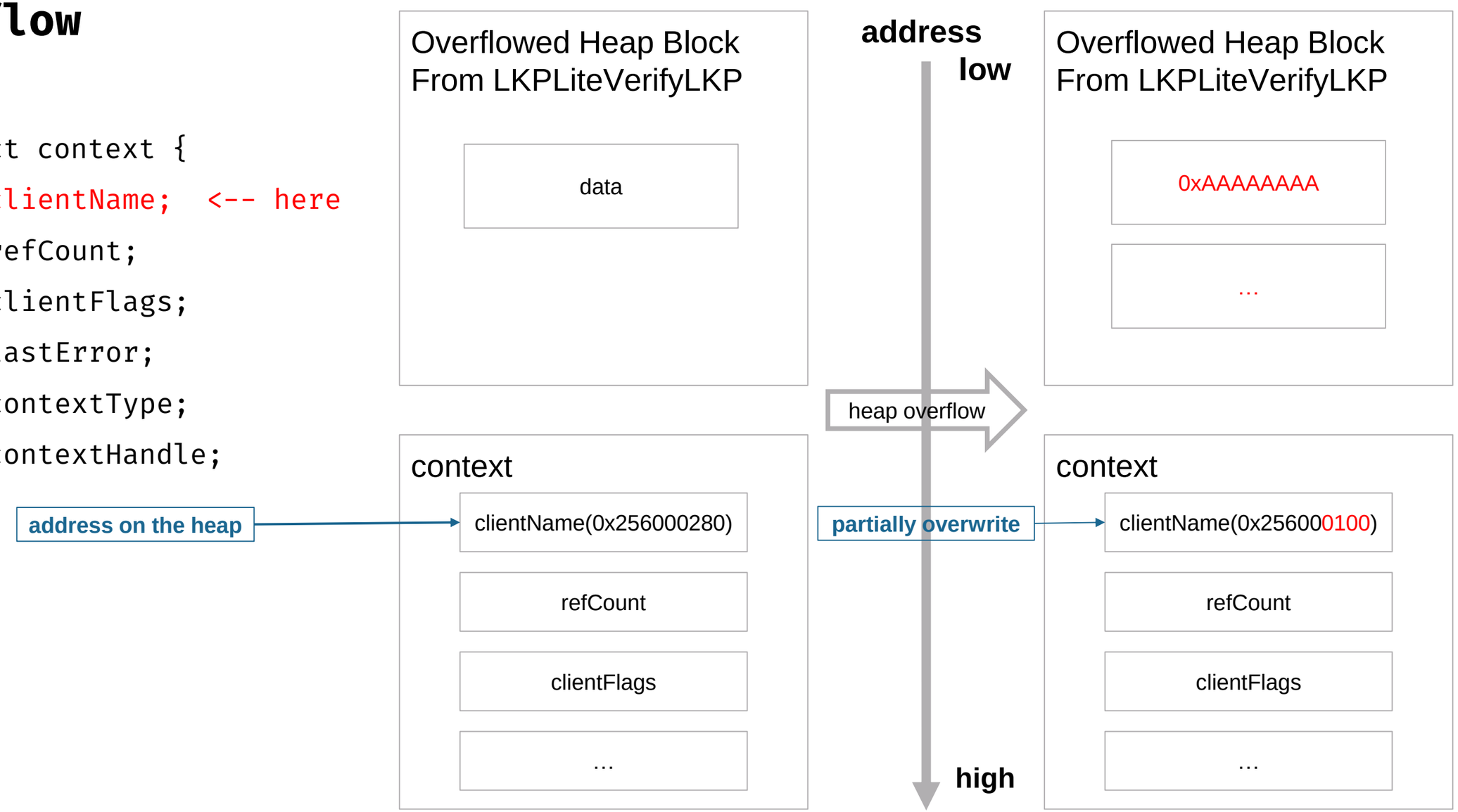
TLSRpcRetrieveTermServCert



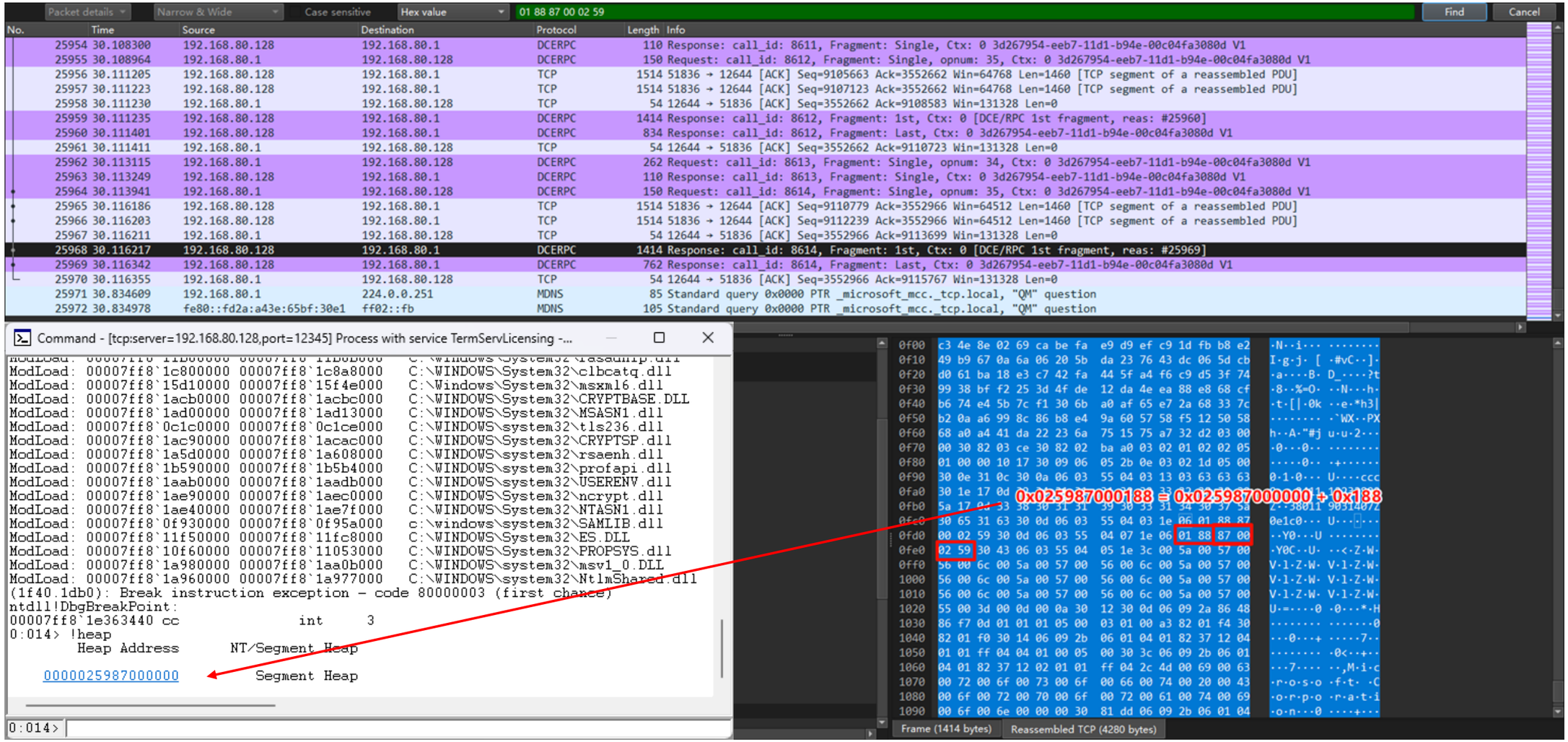
192.168.80.10

Heap Overflow

```
typedef struct context {
    LPTSTR  clientName;  <-- here
    DWORD   refCount;
    DWORD   clientFlags;
    DWORD   lastError;
    DWORD   contextType;
    HANDLE  contextHandle;
}
```



Leak heap address



TLSRpcKeyPackEnumNext

- TLSRpcKeyPackEnumNext

  Need context->contextType == 1

  TLSDBLicenseKeyPackEnumNext

  -> TLSDBKeyPackDescFind

  -> vtable call

- Hijack vtable call

  Modify context->contextType to 1

  Modify context->contextHandle to fake handle

```
TLSRpcRequestTermServCert(
    [in] PCONTEXT_HANDLE phContext,
    [in] LPLSKeyPack lpKeyPack,
    [in, out, ref] PDWORD pdwErrCode
);
```

```
typedef struct context {
    LPTSTR  clientName;
    DWORD   refCount;
    DWORD   clientFlags;
    DWORD   lastError;
    DWORD   contextType;
    HANDLE  contextHandle;
}
```

Constructing Fake Object

- Find functions that can parse our input and save it to the heap

  TLSRpcRegisterLicenseKeyPack

  TLSRpcSendServerCertificate

  ...

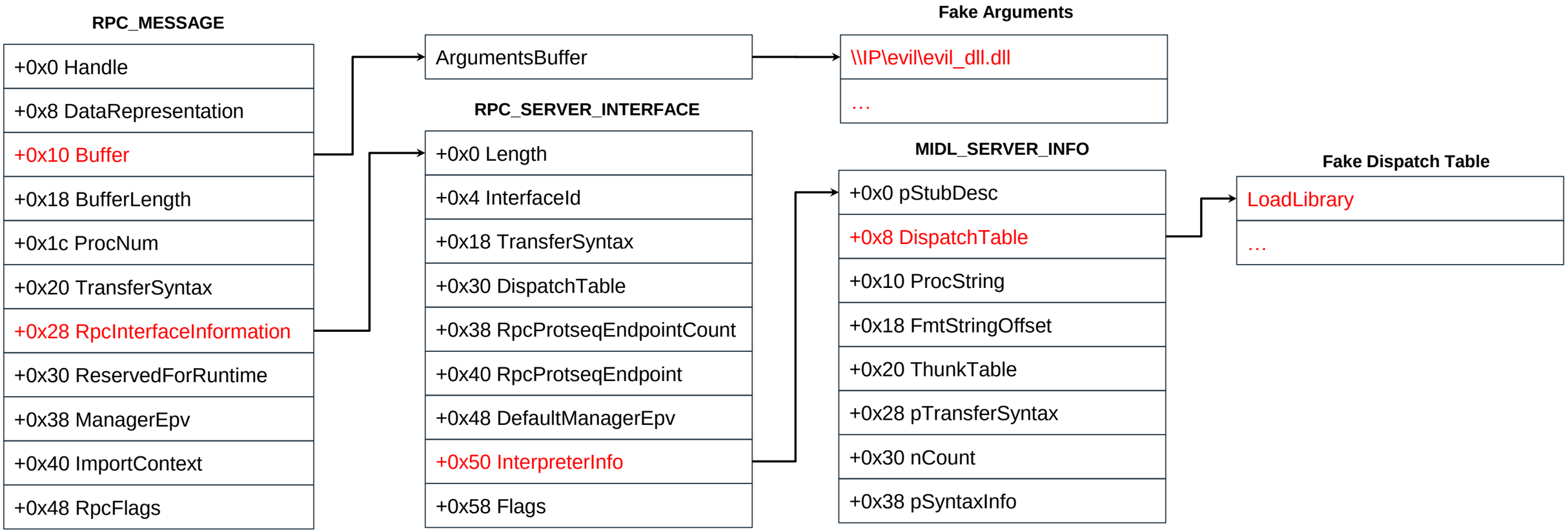
- Heap spraying

  Construct multiple fake objects

  Improve the accuracy of predicted addresses

Bypass CFG

- Hijack Rip to rpcrt4!NdrServerCall2(PRPC_MESSAGE pRpcMsg)



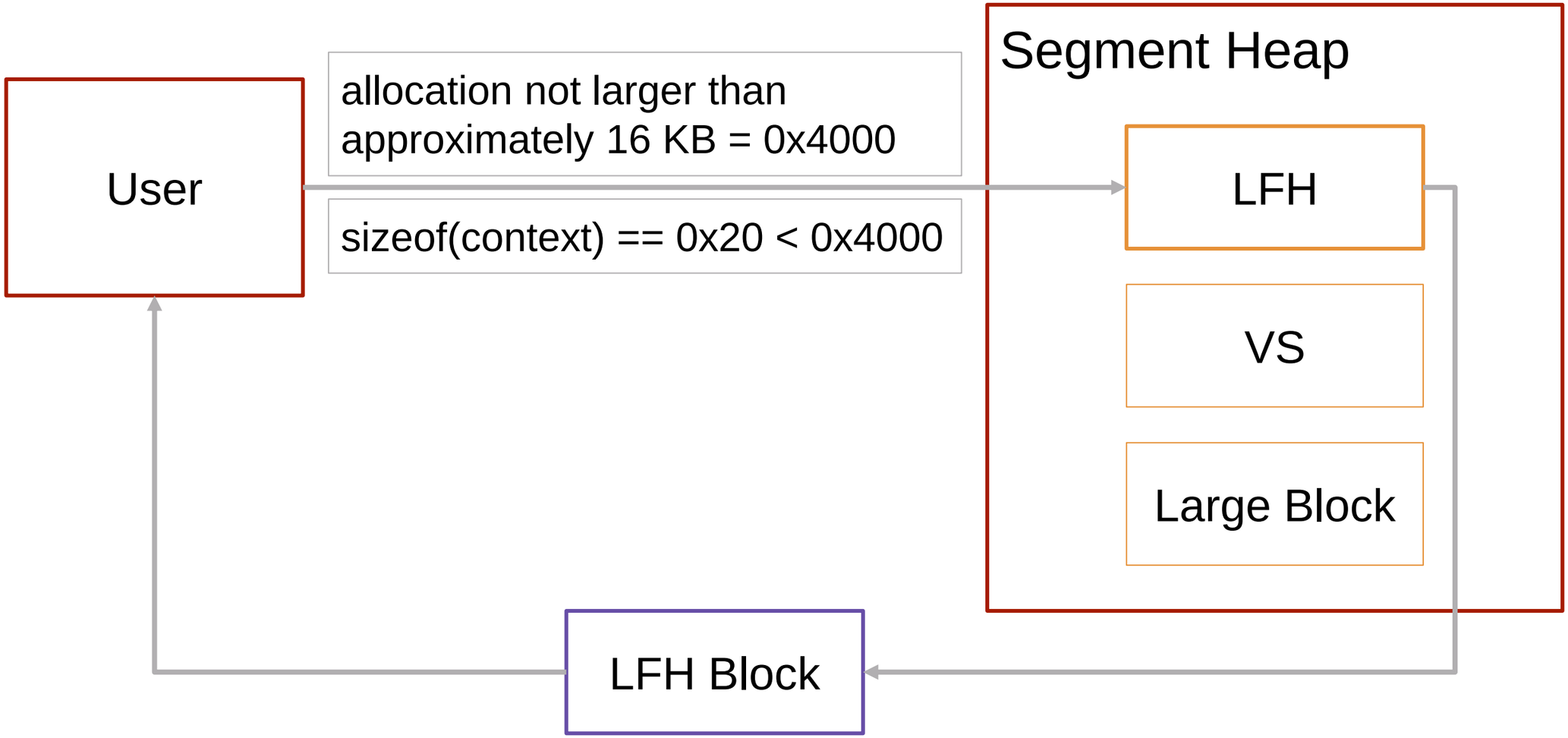
- Also feasible to allocate and execute shellcode in memory

Next

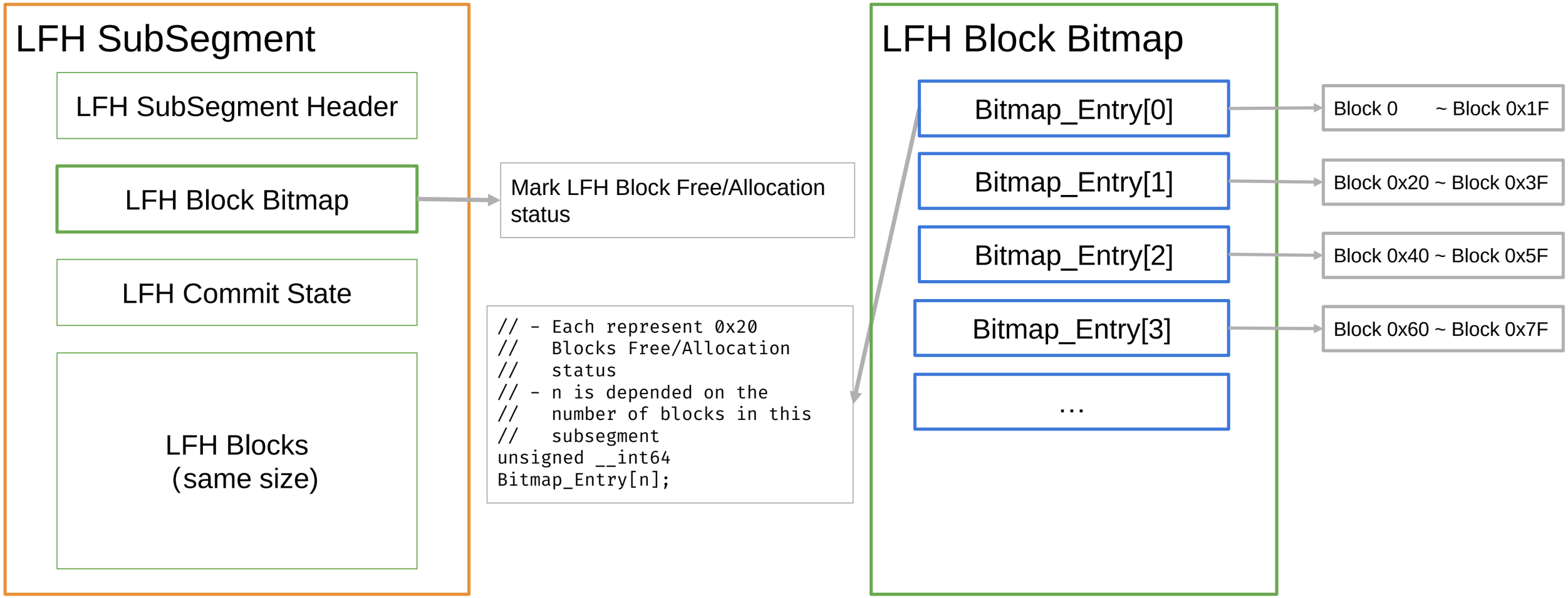
- The exploitation seems really simple!
- but ... really?

Playing with the LFH

Low Fragmentation Heap

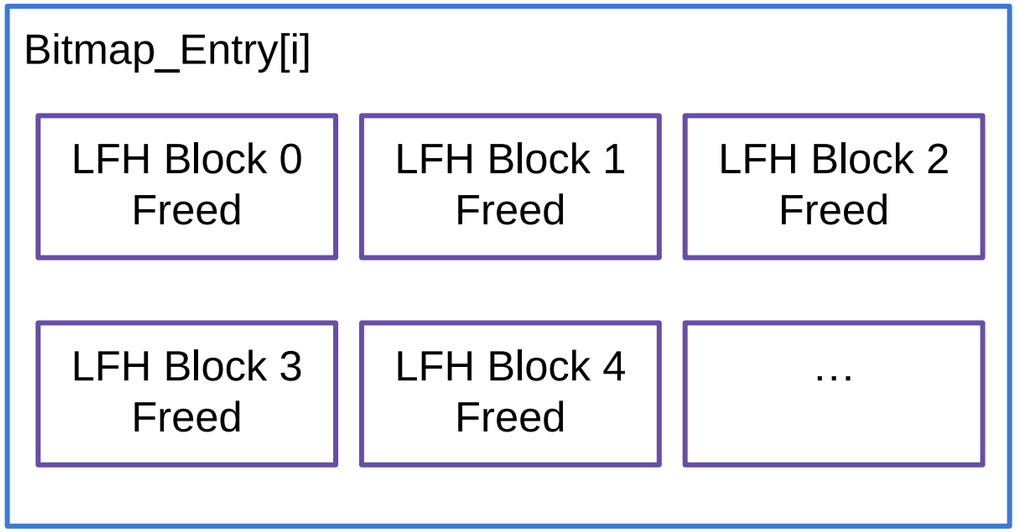


Low Fragmentation Heap

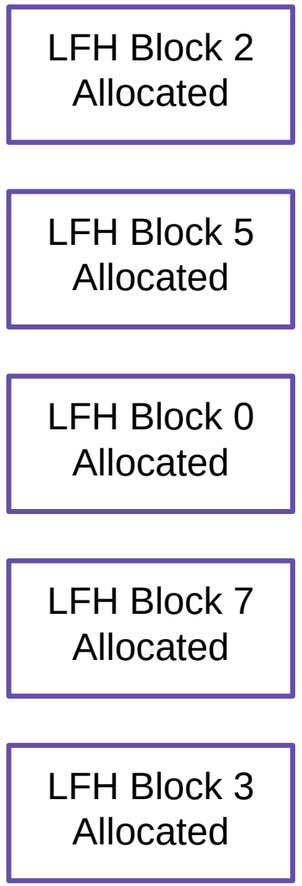


Low Fragmentation Heap

- LFH Block Random Allocation



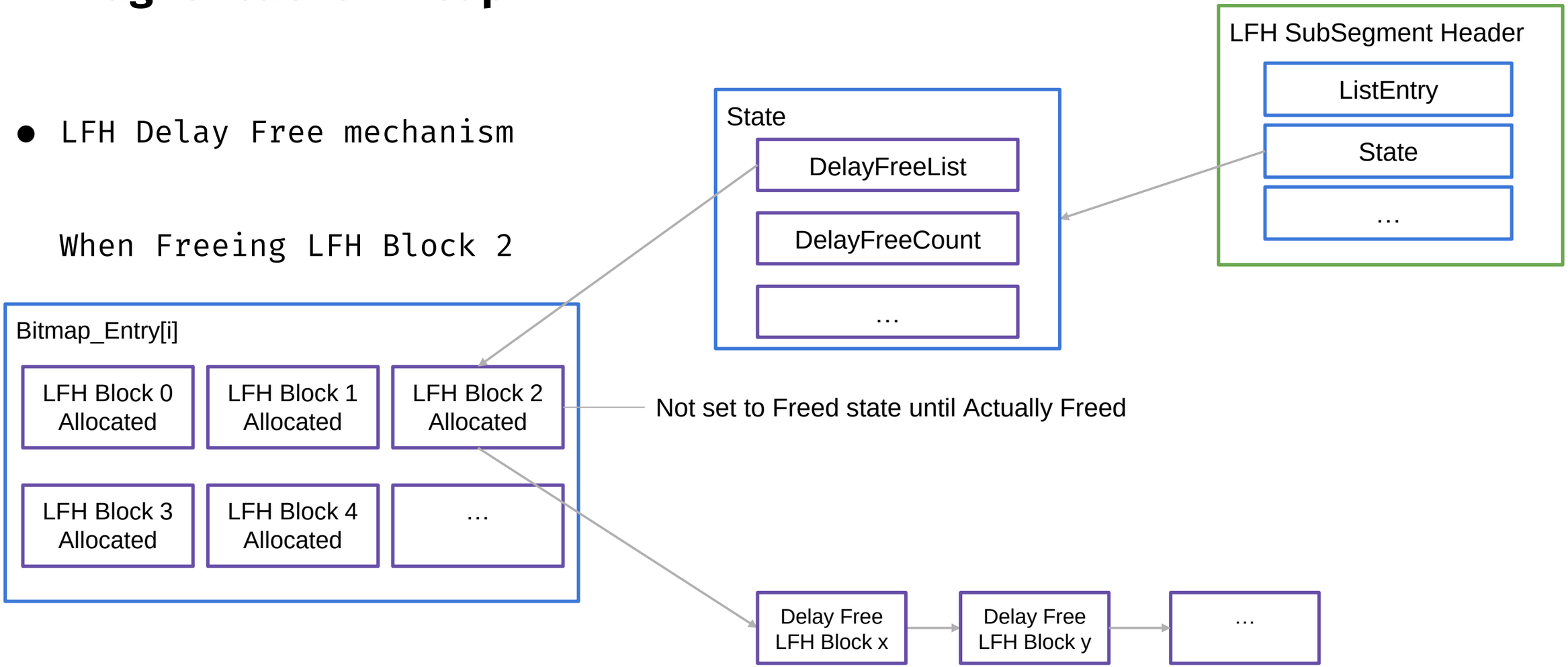
Random Sequence



Low Fragmentation Heap

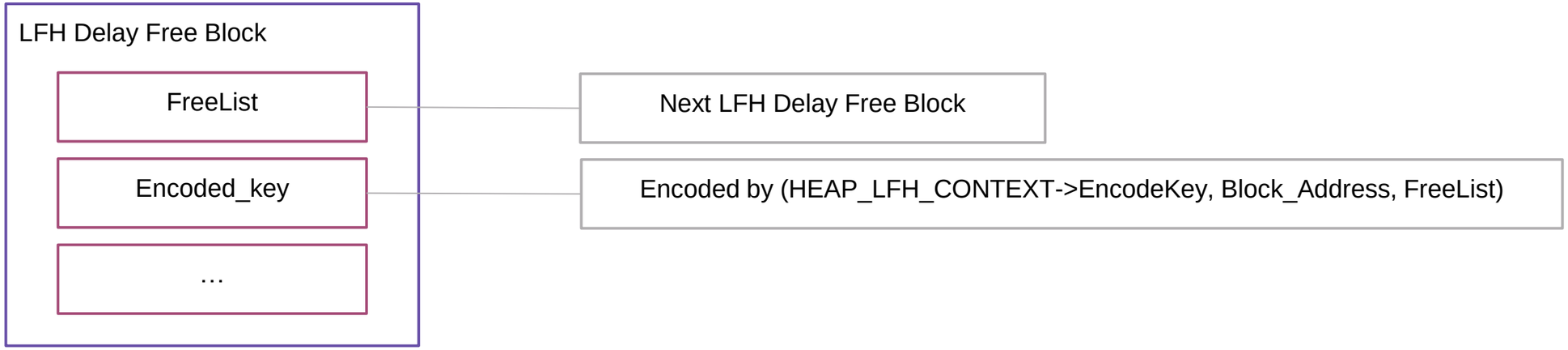
- LFH Delay Free mechanism

When Freeing LFH Block 2



Low Fragmentation Heap

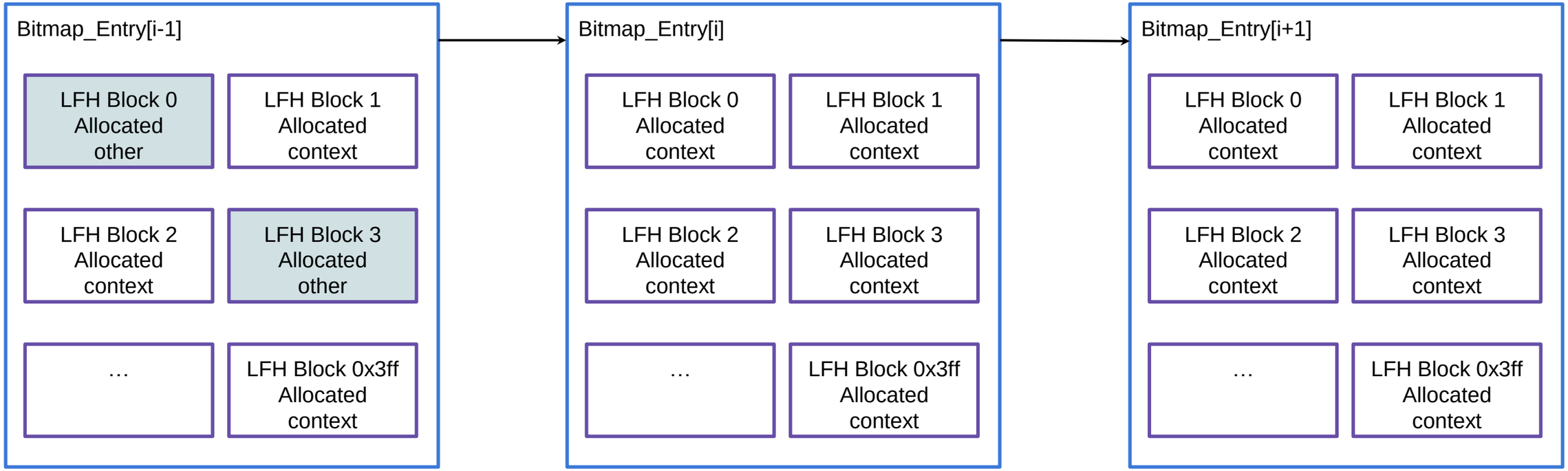
- LFH header encoding



Encoded_key is checked when the Delay Free Block actually gets freed

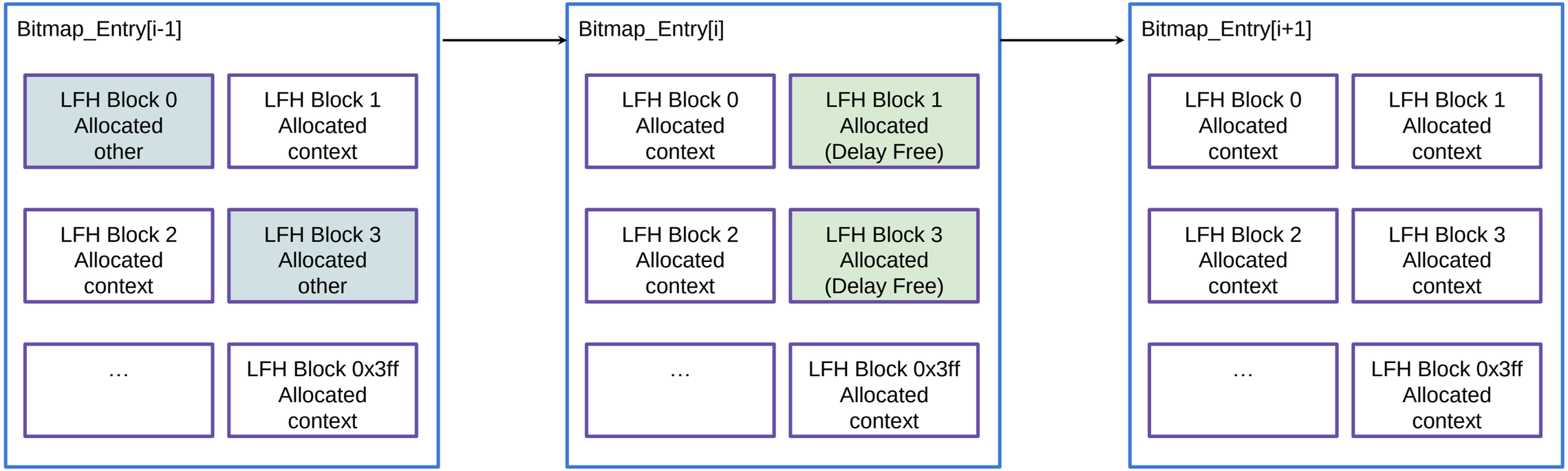
Heap Spray

- **Context Allocation**
- Allocate enough contexts
- Fill up Bitmap_Entry[i] with contexts(No other block)



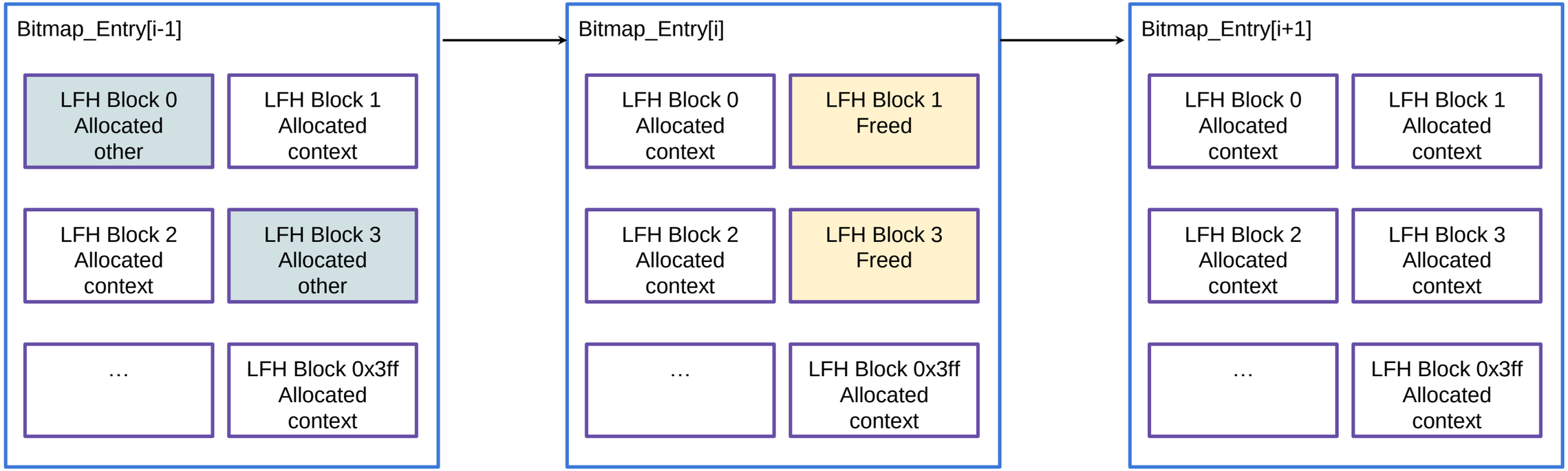
Heap Spray

- **Creating Holes**
- Free small number contexts
- Dig hole in Bitmap_Entry[i] where all the blocks in it is context



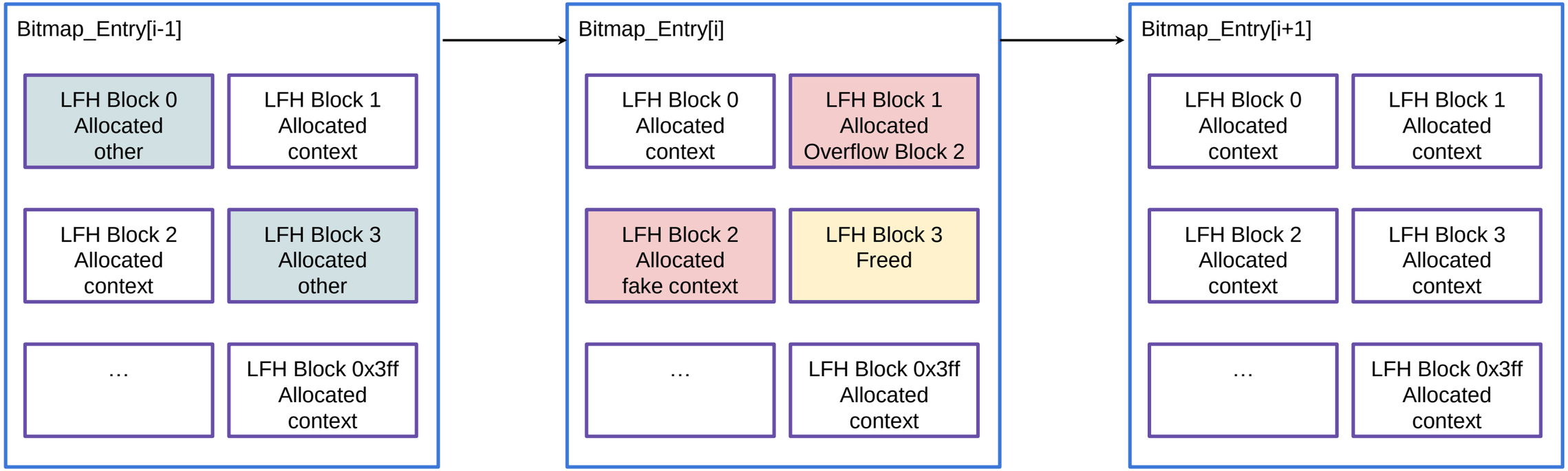
Heap Spray

- **Delay-Free Consideration**
- Wait for a while until delay free blocks actually get freed



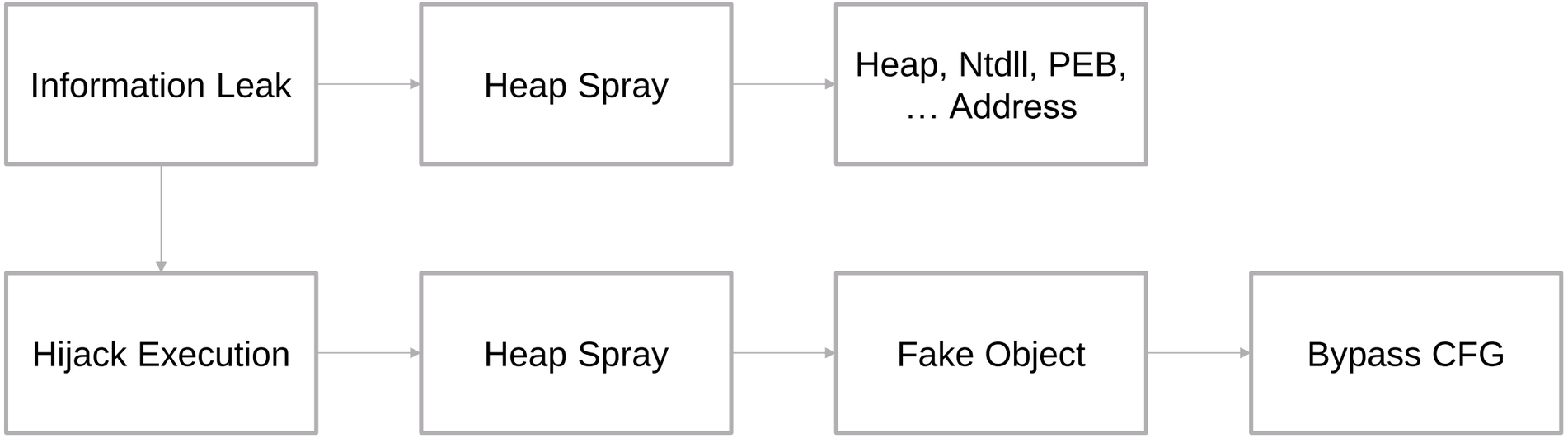
Heap Spray

- **Allocating the Overflow Block**
- Trigger CVE-2024-38077 heap overflow



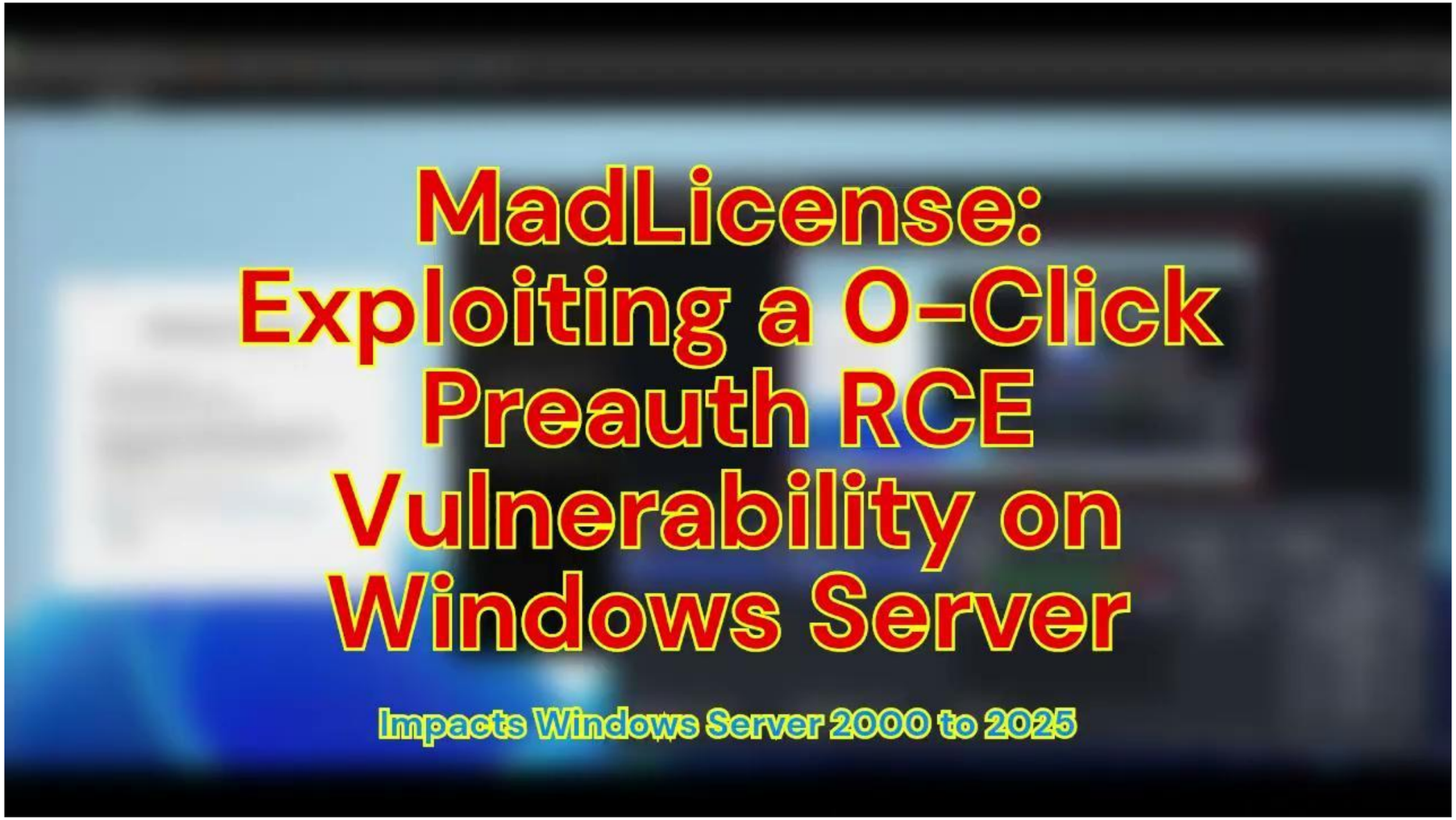
Exploitation Summary

- Exploitation flow chart



Demo：

Success Rate Nearly 100%



Summary

Summary

- Introduction of Remote Desktop Licensing Service
  - Component of Remote Desktop Service
- CVE-2024-38077 we found in RDL
  - Preath Heap Overflow
- Exploitation using CVE-2024-38077
  - Leak address
  - Hijack Rip and Bypass CFG
  - Technique to play with the LFH

Summary

- Highlights

  Based on the Windows Server 2025

  All mitigation enabled

  One bug to rule them all

- Security reinforcements in Windows killing countless vulnerability exploit paths in the last 25 years

  But not all of them are dead

But in Madlicense, the same old heap overflow from 1990s beats Windows Server (2025) Again

Thanks!