



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



APRIL 3-4, 2025
BRIEFINGS

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

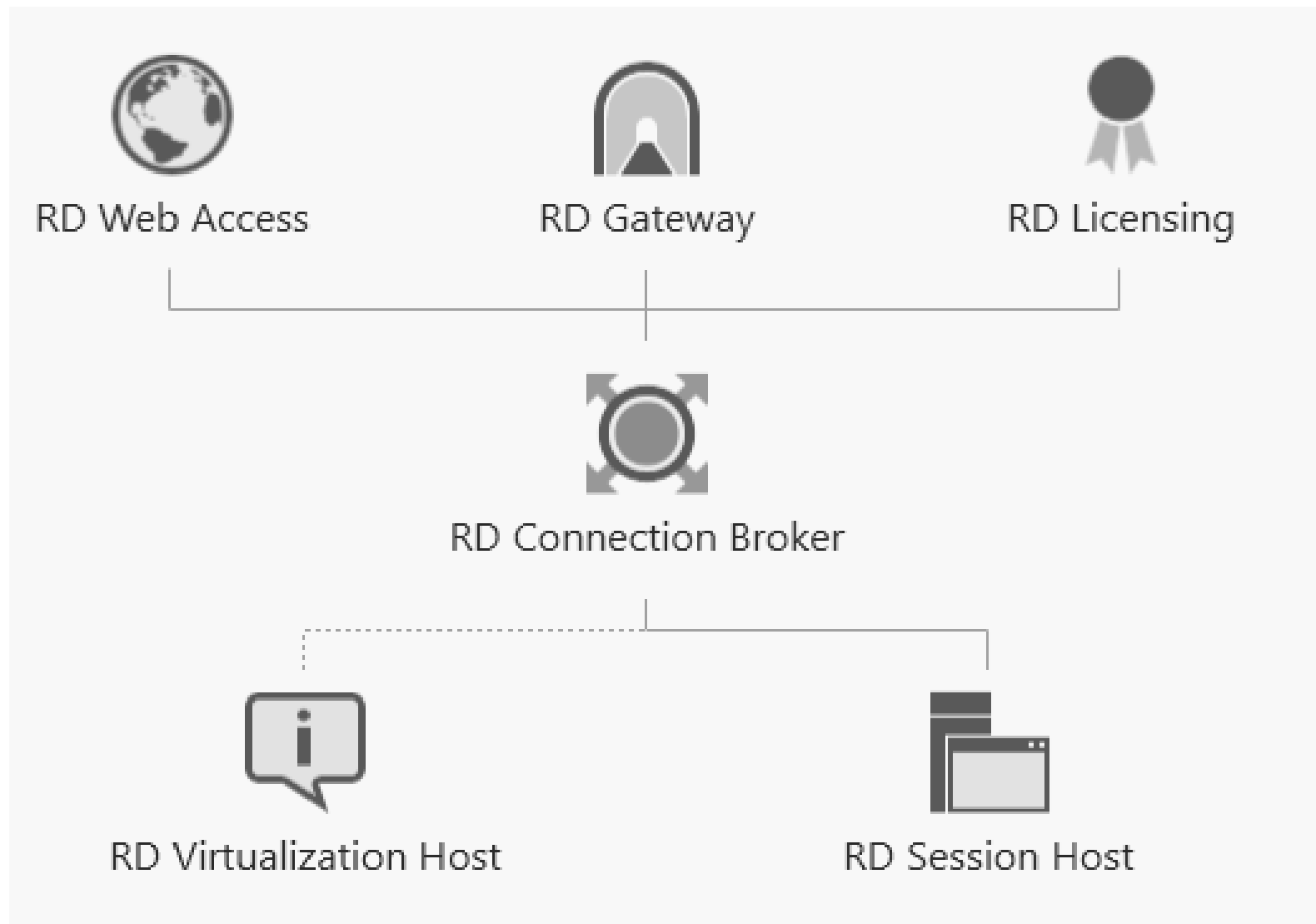
- Windows Remote Desktop License Server Preauth RCE

- One bug to Rule Them All

Remote Desktop Service

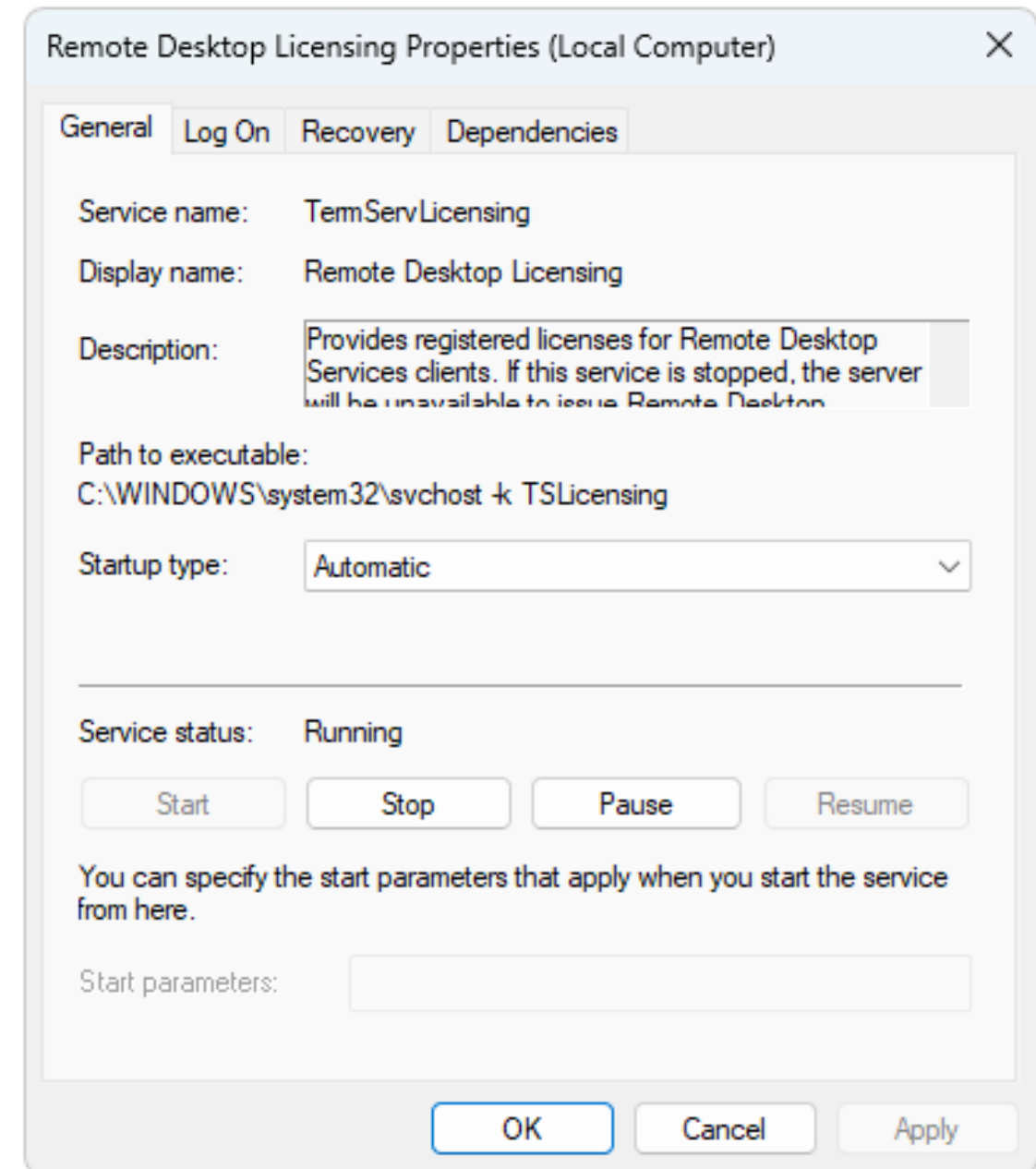
- Remote Desktop Service is a core service on Windows
 - Personal Remote Access
 - Server Management
 - Remote Desktop Rental Services
- Many historical security vulnerabilities
 - From CVE-2012-0002 to CVE-2024-43599
 - No remote exploit for many years

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



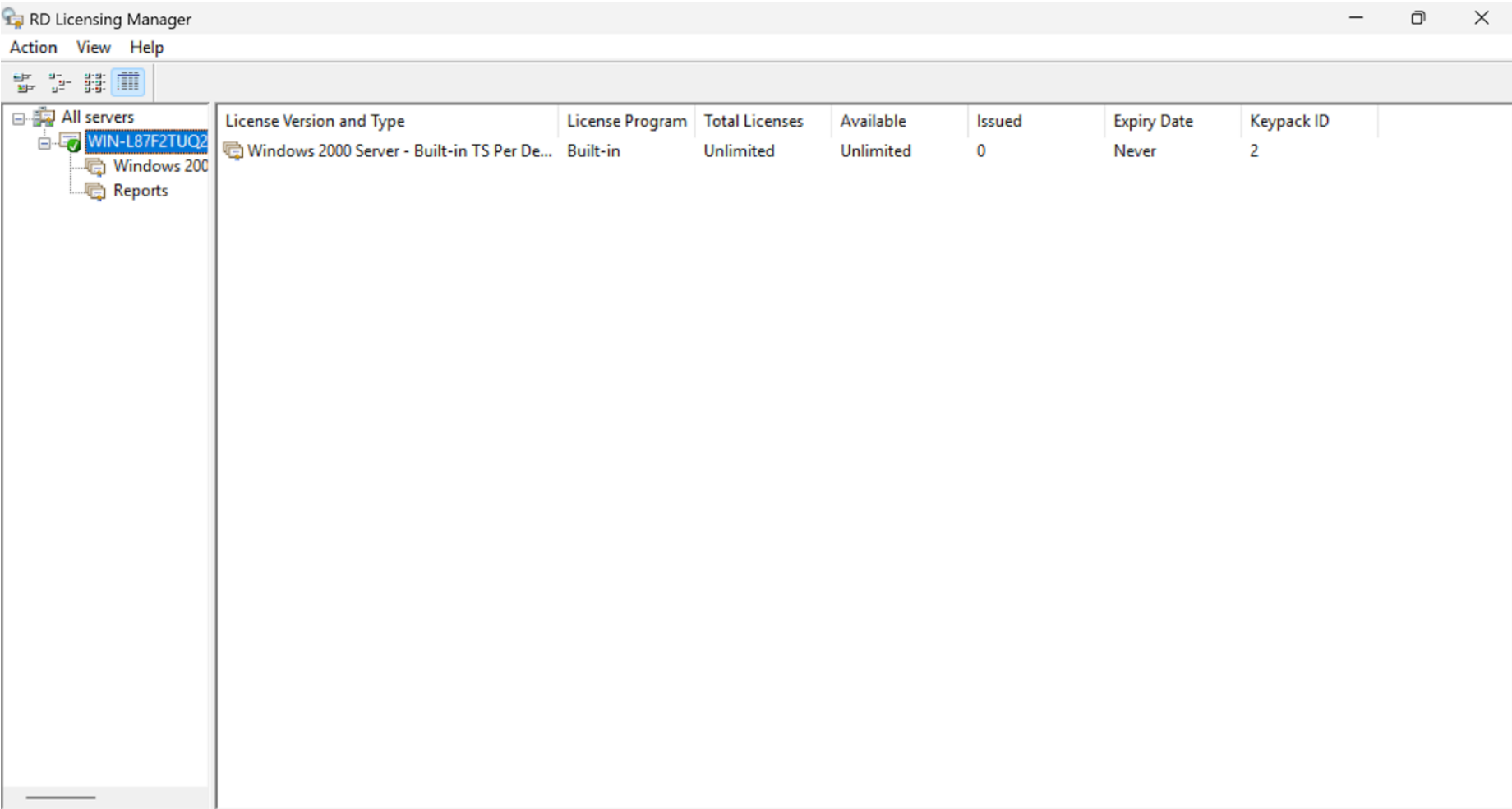


APRIL 3-4, 2025
BRIEFINGS

Remote Desktop License Service RPC Internals

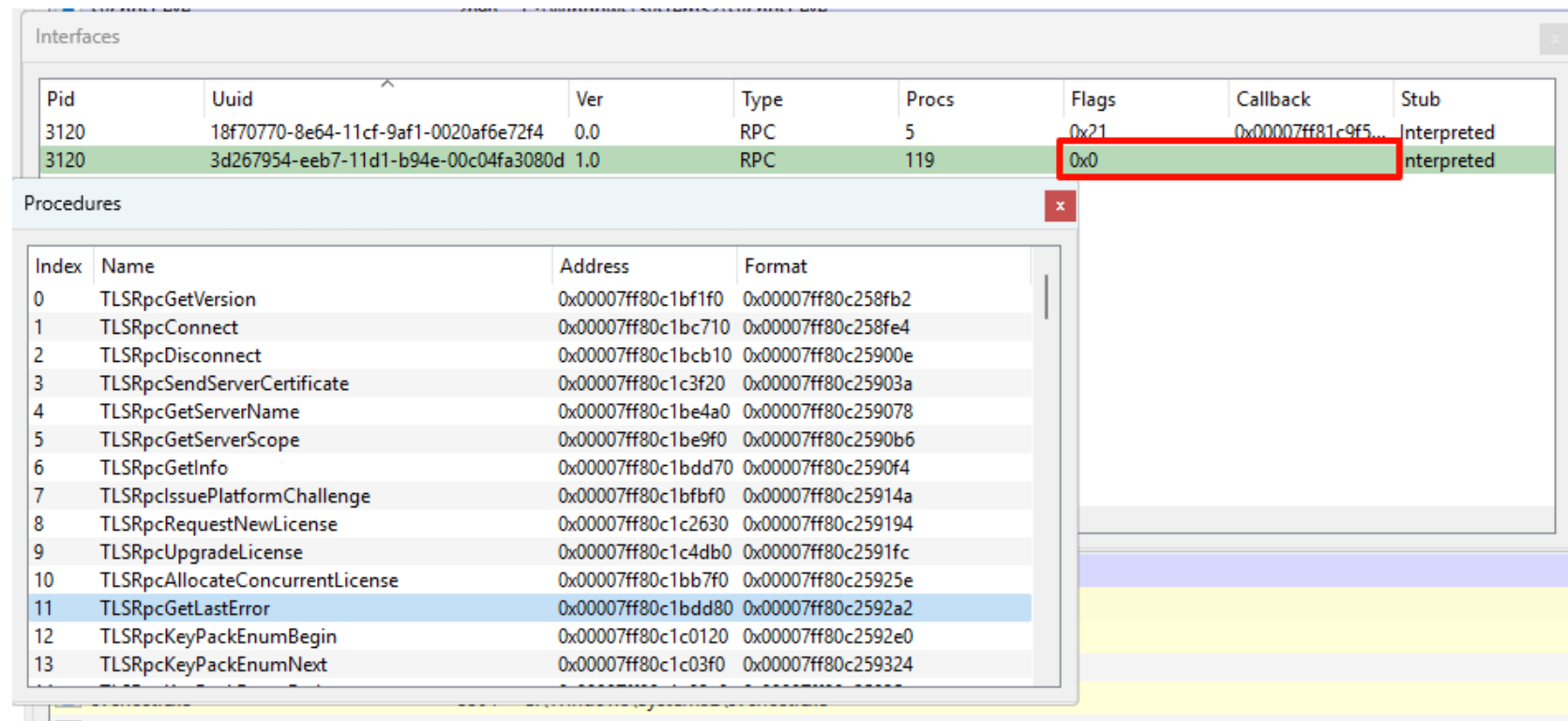
Remote Desktop License Service Internals

- 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



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

Index	Name	Address	Format
0	TLSPcGetVersion	0x00007ff80c1bf1f0	0x00007ff80c258fb2
1	TLSPcConnect	0x00007ff80c1bc710	0x00007ff80c258fe4
2	TLSPcDisconnect	0x00007ff80c1bcb10	0x00007ff80c25900e
3	TLSPcSendServerCertificate	0x00007ff80c1c3f20	0x00007ff80c25903a
4	TLSPcGetServerName	0x00007ff80c1be4a0	0x00007ff80c259078
5	TLSPcGetServerScope	0x00007ff80c1be9f0	0x00007ff80c2590b6
6	TLSPcGetInfo	0x00007ff80c1bdd70	0x00007ff80c2590f4
7	TLSPcIssuePlatformChallenge	0x00007ff80c1bfbf0	0x00007ff80c25914a
8	TLSPcRequestNewLicense	0x00007ff80c1c2630	0x00007ff80c259194
9	TLSPcUpgradeLicense	0x00007ff80c1c4db0	0x00007ff80c2591fc
10	TLSPcAllocateConcurrentLicense	0x00007ff80c1bb7f0	0x00007ff80c25925e
11	TLSPcGetLastError	0x00007ff80c1bdd80	0x00007ff80c2592a2
12	TLSPcKeyPackEnumBegin	0x00007ff80c1c0120	0x00007ff80c2592e0
13	TLSPcKeyPackEnumNext	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)

```
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  
);
```

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





APRIL 3-4, 2025
BRIEFINGS

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  
);
```

Activate Server Wizard

License Server Activation
Enter the license server ID.

To activate your license server, please call Microsoft at this number:

0800 000 000 or Intl. + 43 1795 673 56

For the most current list of telephone numbers, see <http://go.microsoft.com/fwlink/?LinkID=122174>.

You will need your Product ID to complete this operation. Your Product ID is:

00454-40000-58511-AT122

The Microsoft Customer Support representative will provide you with a license server ID. Type this ID in the following boxes.

--	--	--	--	--	--	--

< Back Next > Cancel

CVE-2024-38077

- Heap overflow in CDataCoding::DecodeData

TLSPcTelephoneRegisterLKP -> 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!TLSPcTelephoneRegisterLKP+0x15b
05 000000e5`1737ead0 00007ff8`1d3374ed RPCRT4!Invoke+0x73
06 000000e5`1737eb30 00007ff8`1d336f6a RPCRT4!NdrStubCall12+0x30d
07 000000e5`1737ede0 00007ff8`1d3a3897 RPCRT4!NdrServerCall12+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
```




APRIL 3-4, 2025
BRIEFINGS

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

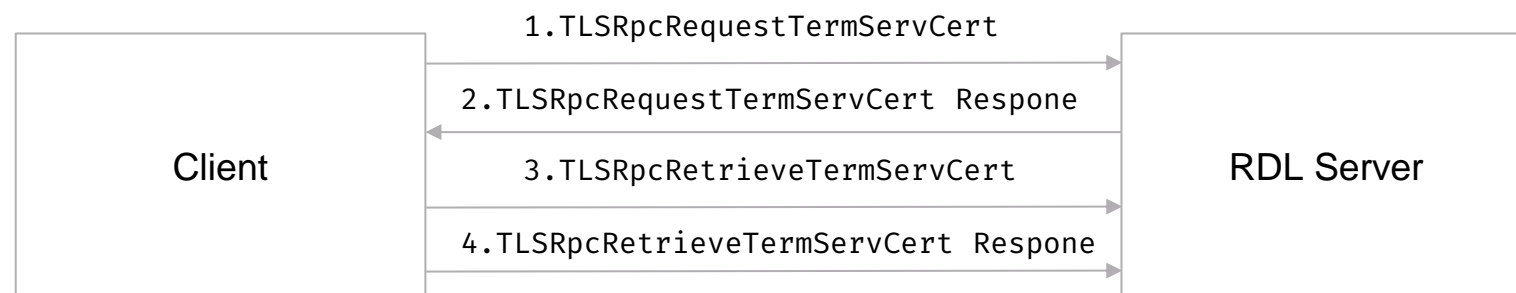
- TLSRpcRequestTermServCert

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

Frame 29: 1414 bytes on wire (11312 bits), 1414 bytes captured (11312 bits) on interface 0
Ethernet II, Src: VMware_48:9a:20 (00:0c:29:48:9a:20), Dst: VMware_c0:00:01 (00:50:56:c0:00:01)
Internet Protocol Version 4, Src: 192.168.80.128, Dst: 192.168.80.1
Transmission Control Protocol, Src Port: 49674, Dst Port: 10207, Seq: 3363, Ack: 602, Len: 1360
[3 Reassembled TCP Segments (4280 bytes): #26(1460), #27(1460), #29(1360)]
Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Response, Fragment: 1st, FragLen: 4280, Call: 5, Ctx: 0, [Req: #2]

0F20 d0 61 ba 18 e3 c7 42 fa 44 5f a4 f6 c9 d5 3f 74 .a....B. D....?t
0F30 99 38 bf f2 25 3d 4f de 12 da 4e ea 88 e8 68 cf .8..%-0. .N...h.
0F40 b6 74 e4 5b 7c f1 30 6b a0 af 65 e7 2a 68 33 7c .t.[].0k .e.*h3]
0F50 b2 0a a6 99 8c 86 b8 e4 9a 60 57 58 f5 12 50 58`WX.PX
0F60 68 a0 a4 41 da 22 23 6a 75 15 75 a7 32 1a 04 00 h..A..#j u..u.2...
0F70 00 30 82 04 16 30 82 03 02 a0 03 02 01 02 02 05 .0..0..
0F80 01 00 00 00 05 30 09 06 05 2b 0e 03 02 1d 05 000.. +.....
0F90 30 0e 31 0c 30 0a 06 03 55 04 03 13 03 63 63 63 0.1.0.. U....ccc
0Fa0 30 1e 17 0d 32 34 31 31 31 32 32 32 32 31 36 0...2411 12222216
0Fb0 5a 17 0d 33 38 30 31 31 39 30 33 31 34 30 37 5a Z..38011 90314072
0Fc0 30 81 ac 31 81 a9 30 27 06 03 55 04 03 1e 20 00 .0..1..0' ..U...
0Fd0 6e 00 63 00 61 00 63 00 6e 00 5f 00 69 00 70 00 n..c..a..c. n..i..p.
0Fe0 5f 00 74 00 63 00 70 00 3a 00 31 00 39 00 32 30 .0...f..0..0..0..0..
0Ff0 39 06 03 55 04 07 1e 32 00 6e 00 63 00 61 00 63 9..U..2..n.c.a.c
1000 00 6e 00 5f 00 69 00 70 00 5f 00 74 00 63 00 70 .n..i..p. .t..c..p
1010 00 3a 00 31 00 39 00 32 00 2e 00 31 00 36 00 38 ..:1.9.2 ..:1.6.8
1020 00 2e 00 38 00 30 00 2e 00 31 30 43 06 03 55 04 ..:8.0.. .10C..U..
1030 05 1e 3c 00 5a 00 57 00 56 00 6c 00 5a 00 57 00 ...7.W. V.1.7.W..
1040 56 00 6c 00 5a 00 57 00 56 00 6c 00 5a 00 57 00 V.1.2.W. V.1.2.W..
1050 56 00 6c 00 5a 00 57 00 56 00 6c 00 5a 00 57 00 V.1.2.W. V.1.2.W..
1060 56 00 6c 00 5a 00 57 00 55 00 3d 00 0d 00 0a 30 V.1.2.W. U=....0
1070 12 30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05 00 .0...*H
1080 03 01 00 a3 82 01 f4 30 82 01 f0 30 14 06 09 2b0+
1090 06 01 04 01 82 37 12 04 01 01 ff 04 04 01 00 057..
10a0 00 30 3c 06 09 2b 06 01 04 01 82 37 12 02 01 01 .0c...+...7....
10b0 ff 04 2c 4d 00 69 00 63 ..,M..i..c

192.168.80.10

Heap Overflow

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

address on the heap

Overflowed Heap Block
From LKPLiteVerifyLKP

data

context

clientName(0x256000280)

refCount

clientFlags

...

address
low

heap overflow

partially overwrite

high

Overflowed Heap Block
From LKPLiteVerifyLKP

0xAAAAAAAA

...

context

clientName(0x256000**100**)

refCount

clientFlags

...

Leak heap address

The image shows a network traffic capture and a debugger window. The network traffic window at the top displays a list of packets. Packet 25968 is highlighted, showing a DCERPC response from 192.168.80.128 to 192.168.80.1. The data field shows a response for call_id 8614, fragment 1st, with context 0 [DCE/RPC 1st fragment, reas: #25969].

The debugger window at the bottom shows the process with service TermServLicensing. The command prompt displays the following output:

```

ModLoad: 00007ff8`1c800000 00007ff8`1c8a8000 C:\WINDOWS\System32\clbcatq.dll
ModLoad: 00007ff8`15d10000 00007ff8`15f4e000 C:\WINDOWS\System32\msxml6.dll
ModLoad: 00007ff8`1ac00000 00007ff8`1acbc000 C:\WINDOWS\System32\CRYPTBASE.DLL
ModLoad: 00007ff8`1ad00000 00007ff8`1ad13000 C:\WINDOWS\System32\MSASN1.dll
ModLoad: 00007ff8`0c1c0000 00007ff8`0c1ce000 C:\WINDOWS\system32\tls236.dll
ModLoad: 00007ff8`1ac90000 00007ff8`1acac000 C:\WINDOWS\System32\CRYPTSP.dll
ModLoad: 00007ff8`1a5d0000 00007ff8`1a608000 C:\WINDOWS\system32\rsaenh.dll
ModLoad: 00007ff8`1b590000 00007ff8`1b5b4000 C:\WINDOWS\system32\profapi.dll
ModLoad: 00007ff8`1aab0000 00007ff8`1aad0000 C:\WINDOWS\system32\USERENV.dll
ModLoad: 00007ff8`1ae90000 00007ff8`1aec0000 C:\WINDOWS\System32\ncrypt.dll
ModLoad: 00007ff8`1ae40000 00007ff8`1ae7f000 C:\WINDOWS\System32\NTASN1.dll
ModLoad: 00007ff8`0f930000 00007ff8`0f95a000 c:\windows\system32\SAMLIB.dll
ModLoad: 00007ff8`11f50000 00007ff8`11fc8000 C:\WINDOWS\System32\ES.DLL
ModLoad: 00007ff8`10f60000 00007ff8`11053000 C:\WINDOWS\System32\PROPSYS.dll
ModLoad: 00007ff8`1a980000 00007ff8`1aa0b000 C:\WINDOWS\system32\msv1_0.DLL
ModLoad: 00007ff8`1a960000 00007ff8`1a977000 C:\WINDOWS\system32\NtLmShared.dll
(1f40.1db0): Break instruction exception - code 80000003 (first chance)
ntdll!DbgBreakPoint:
00007ff8`1e363440 cc          int     3
0:014> !heap
Heap Address      NT/Segment Heap
0000025987000000 ← Segment Heap

```

A red arrow points from the hex value 01888700 in the network traffic data field to the heap address 0000025987000000 in the debugger output.

The hex dump on the right shows the raw data of the packet, with the value 01888700 highlighted in red. A red box around the value 01888700 in the hex dump is also highlighted.

TLSPcKeyPackEnumNext

- TLSPcKeyPackEnumNext

```
Need context->contextType == 1
TlsDBLicenseKeyPackEnumNext
-> TlsDBKeyPackDescFind
-> vtable call
```

- Hijack vtable call

```
Modify context->contextType to 1
Modify context->contextHandle to fake handle
```

```
TLSPcRequestTermServCert(
    [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`

`...`

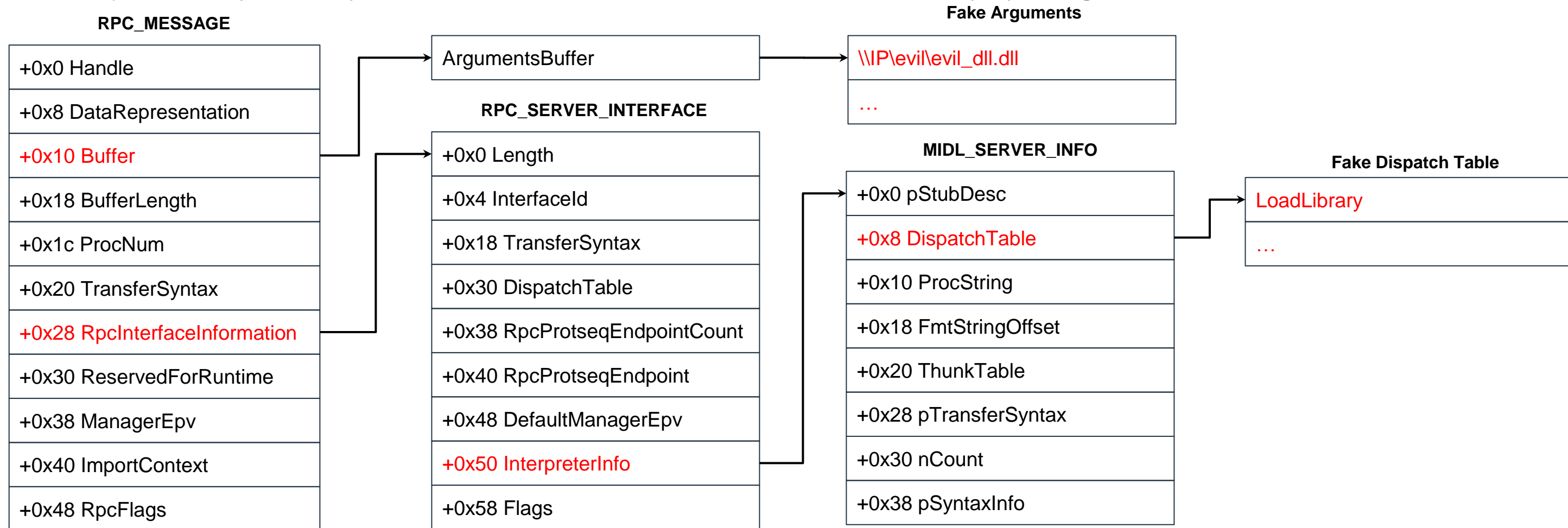
- Heap spraying

`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?

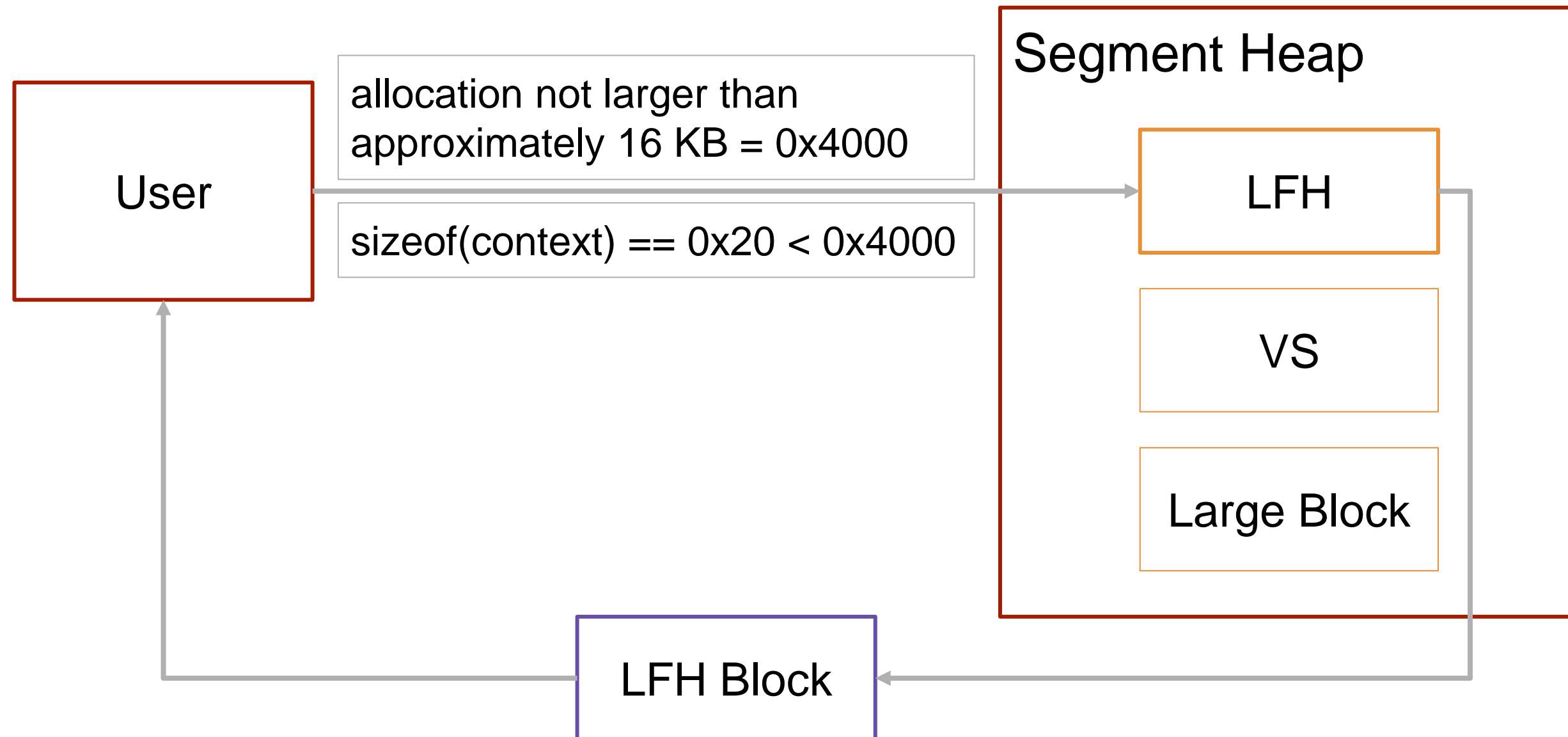




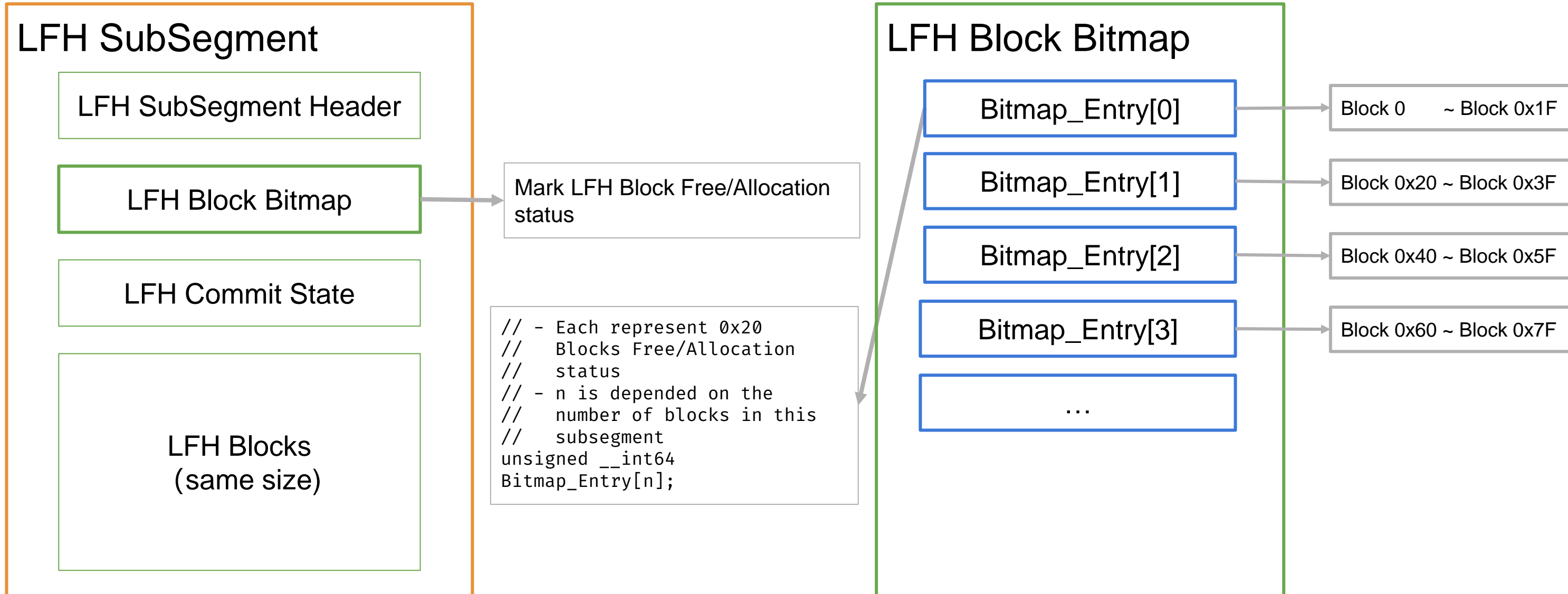
APRIL 3-4, 2025
BRIEFINGS

Playing with the LFH

Low Fragmentation Heap

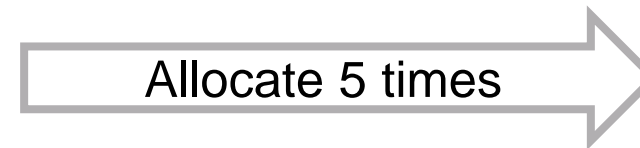
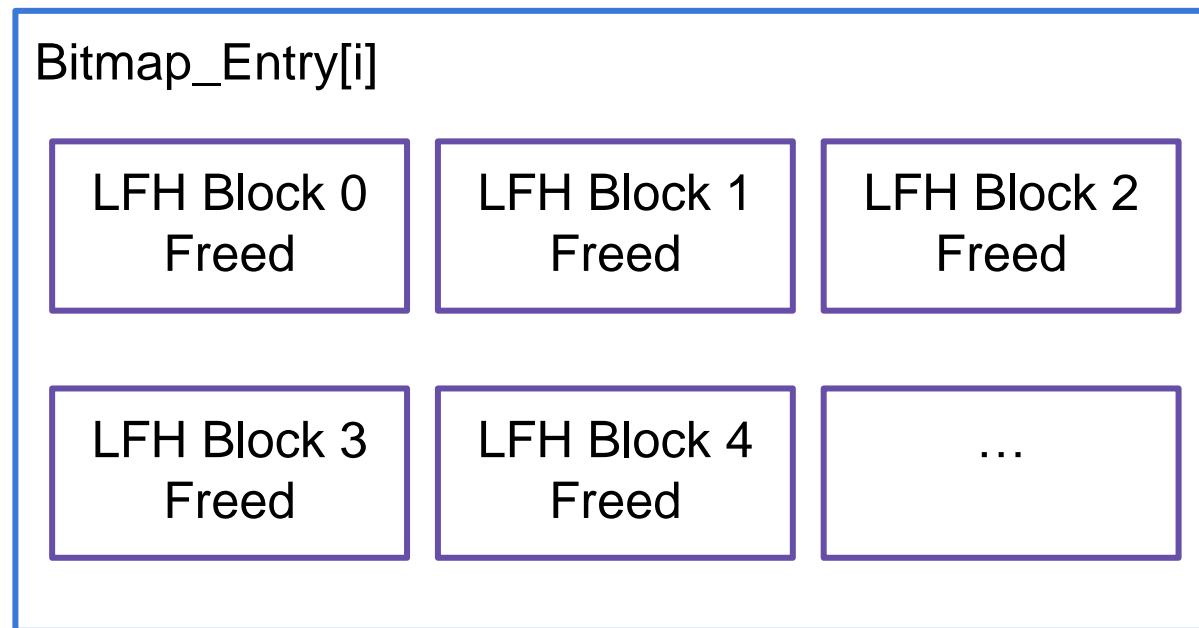


Low Fragmentation Heap



Low Fragmentation Heap

- LFH Block Random Allocation



Random Sequence

LFH Block 2
Allocated

LFH Block 5
Allocated

LFH Block 0
Allocated

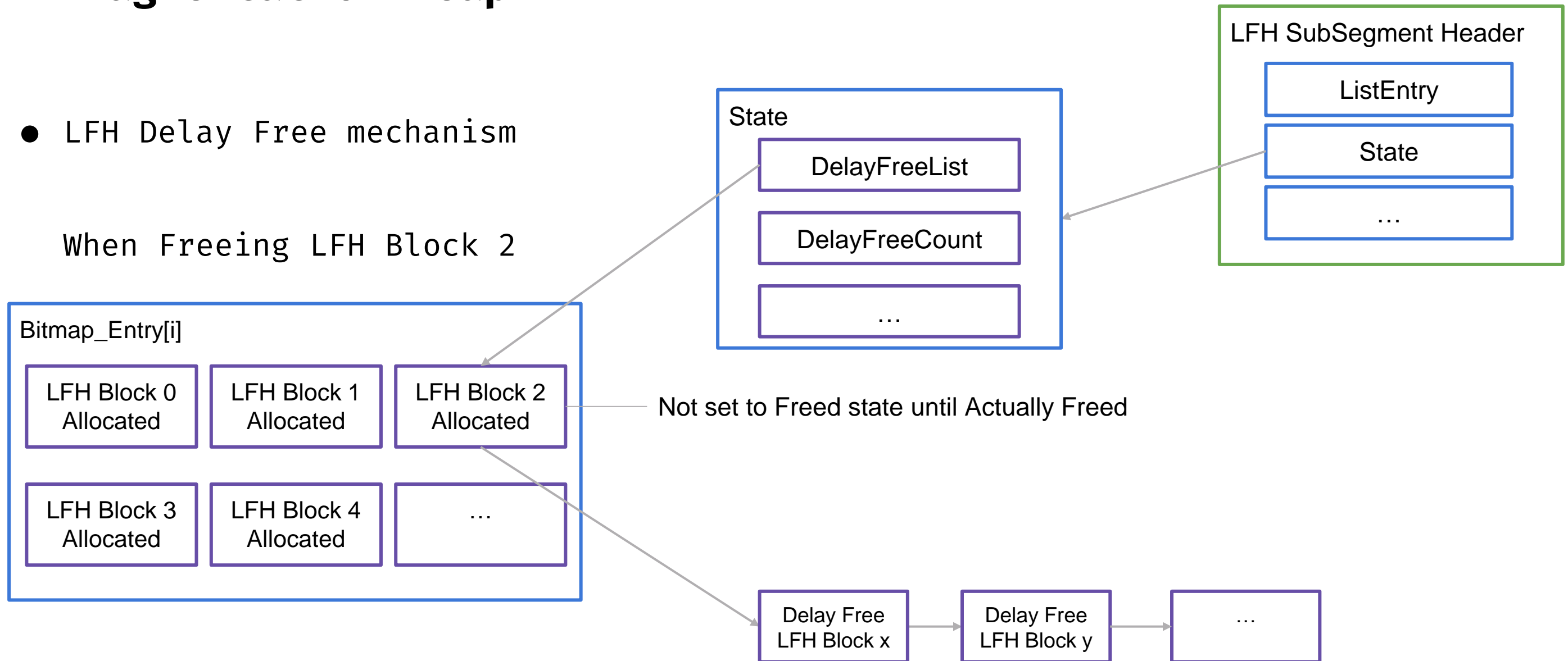
LFH Block 7
Allocated

LFH Block 3
Allocated

Low Fragmentation Heap

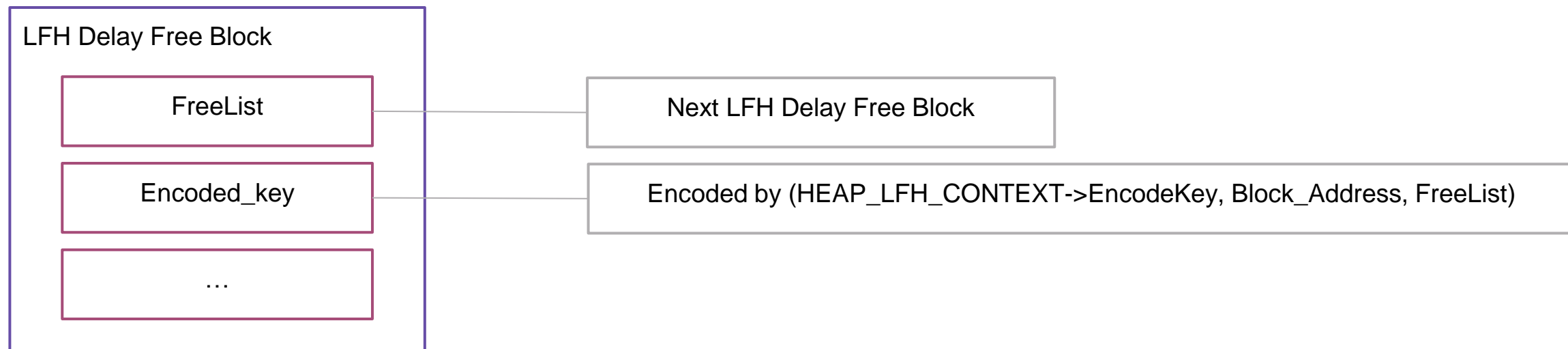
- LFH Delay Free mechanism

When Freeing LFH Block 2



Low Fragmentation Heap

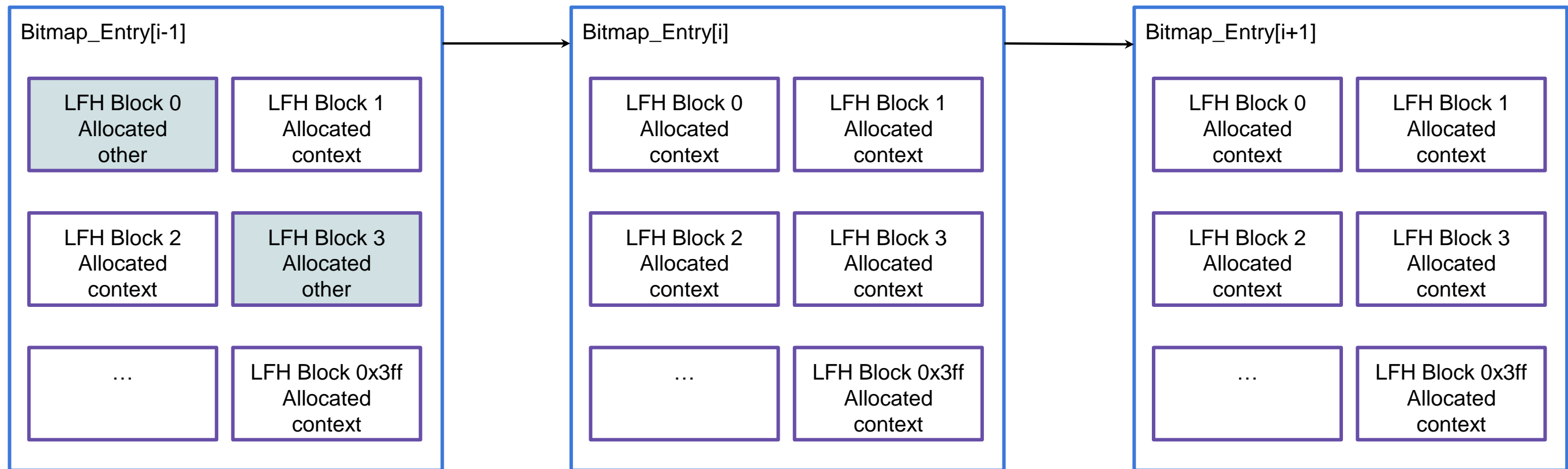
- LFH header encoding



Encoded_key is checked when the Delay Free Block actually gets freed

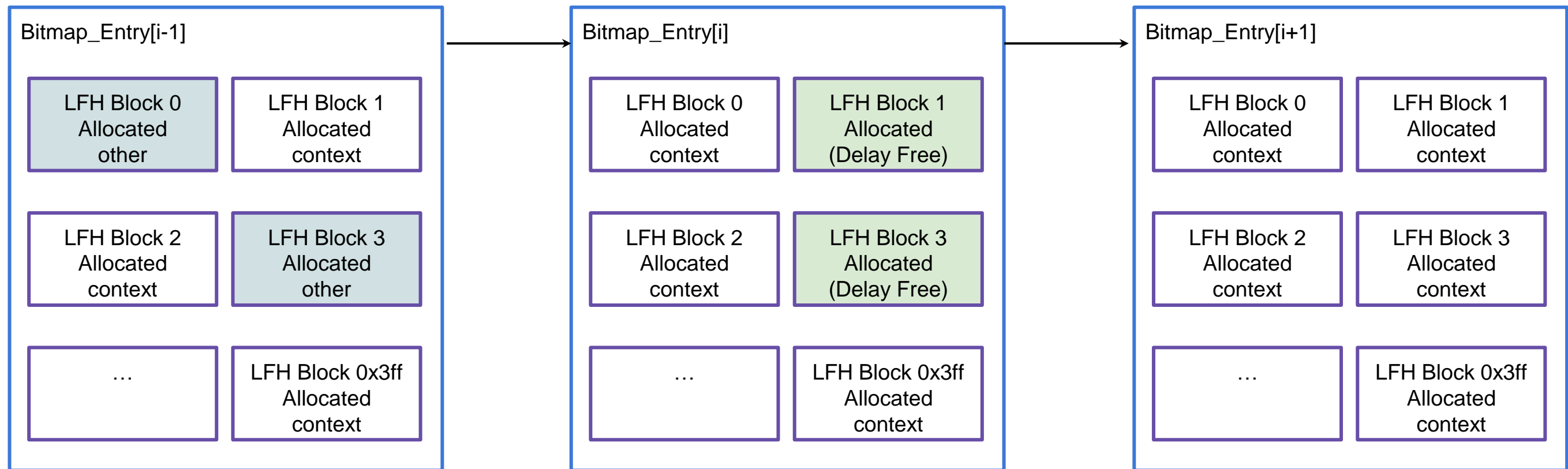
Heap Spray

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



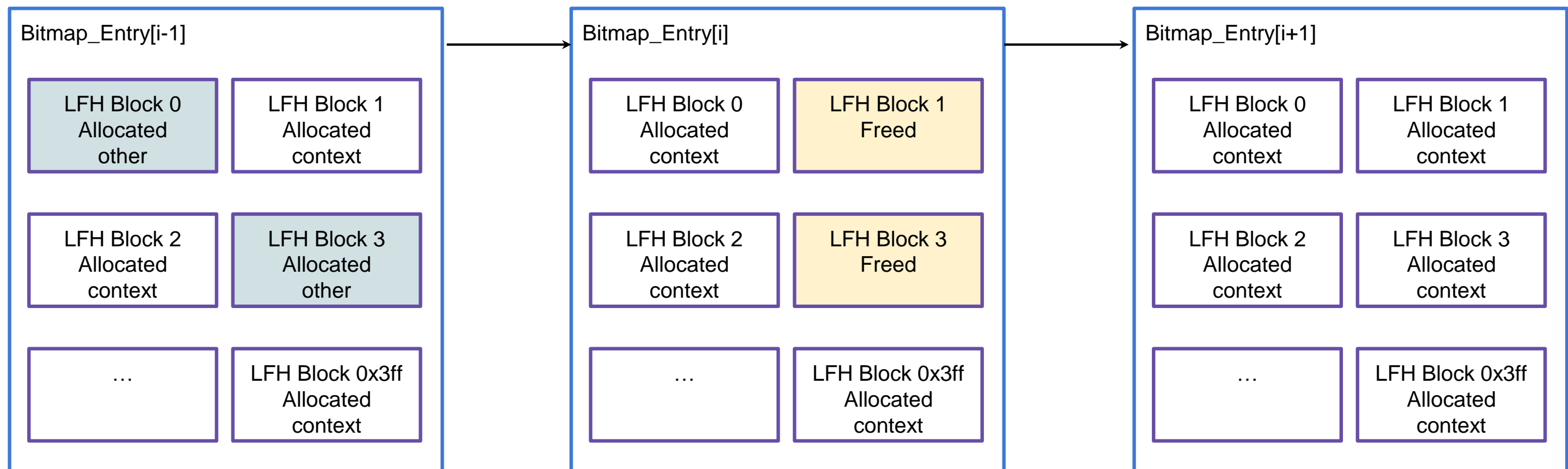
Heap Spray

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



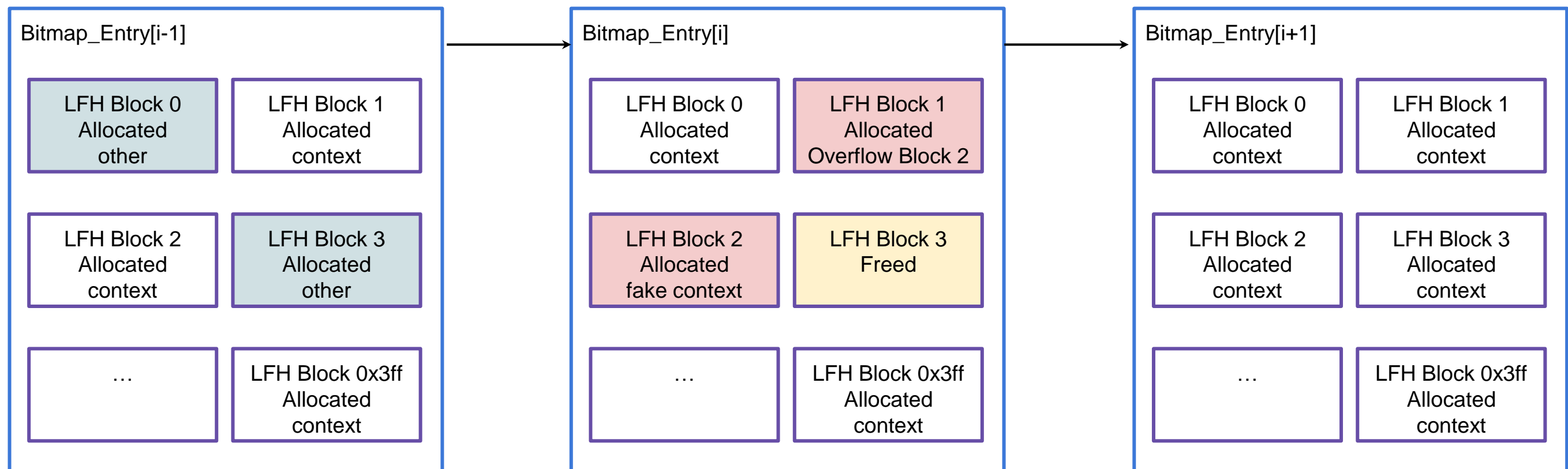
Heap Spray

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



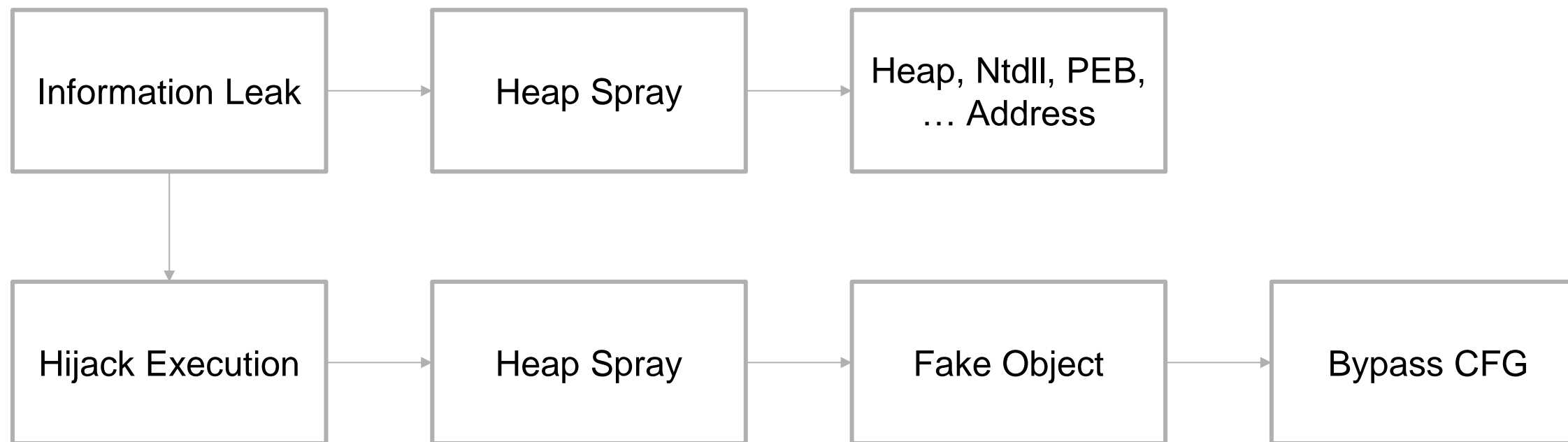
Heap Spray

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



Exploitation Summary

- Exploitation flow chart



Demo :

Success Rate Nearly 100%

MadLicense: Exploiting a 0-Click Preauth RCE Vulnerability on Windows Server

Impacts Windows Server 2000 to 2025



APRIL 3-4, 2025
BRIEFINGS

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



APRIL 3-4, 2025
BRIEFINGS

Thanks!