black hat USA 2023

AUGUST 9-10, 2023

BRIEFINGS

Diving into Windows Remote Access Service for Pre-Auth Bugs

Yuki Chen (@guhe120)

Cyber Kunlun

Whoami

Yuki Chen (@guhe120)

- Bug Hunter & Bug Bounty Lover
- Winner of multiple targets at Pwn2Own 15/16/17, Tianfu Cup 18/19/20
- Four times yearly #1 of MSRC most valuable security researchers
- Won 2 Pwnie Awards – Best RCE and Epic Achievement
- Hardcore ACG Otaku

CYBER KUNLUN

Some of my bugs

CVE-2014-0290,CVE-2014-0321,CVE-2014-1753,CVE-2014-1769,CVE-2014-1782,CVE-2014-1804,CVE-2014-2768,CVE-2014-2802,CVE-2014-2803,CVE-2014-2824,CVE-2014-4057,CVE-2014-4092,CVE-2014-4091, CVE-2014-4095,CVE-2014-4096,CVE-2014-4097,CVE-2014-4082,CVE-2014-4105,CVE-2014-4129,CVE-2014-6369,CVE-2015-0029,CVE-2015-1745,CVE-2015-1743,CVE-2015-3134,CVE-2015-3135,CVE-2015-4431, CVE-2015-5552,CVE-2015-5553,CVE-2015-5559,CVE-2015-6682,CVE-2015-7635,CVE-2015-7636,CVE-2015-7637,CVE-2015-7638,CVE-2015-7639,CVE-2015-7640,CVE-2015-7641,CVE-2015-7642,CVE-2015-7643, CVE-2015-8454,CVE-2015-8059,CVE-2015-8058,CVE-2015-8055,CVE-2015-8057,CVE-2015-8056,CVE-2015-8061,CVE-2015-8067,CVE-2015-8066,CVE-2015-8062,CVE-2015-8068,CVE-2015-8064,CVE-2015-8065, CVE-2015-8063,CVE-2015-8405,CVE-2015-8404,CVE-2015-8402,CVE-2015-8403,CVE-2015-8071,CVE-2015-8401,CVE-2015-8406,CVE-2015-8069,CVE-2015-8070,CVE-2015-8440,CVE-2015-8409,CVE-2015-8047, CVE-2015-8455,CVE-2015-8045,CVE-2015-8441,CVE-2016-0980,CVE-2016-1015,CVE-2016-1016,CVE-2016-1017,CVE-2016-4120,CVE-2016-4160,CVE-2016-4161,CVE-2016-4162,CVE-2016-4163,CVE-2016-4185, CVE-2016-4249,CVE-2016-4180,CVE-2016-4181,CVE-2016-4183,CVE-2016-4184,CVE-2016-4186,CVE-2016-4187,CVE-2016-4233,CVE-2016-4234,CVE-2016-4235,CVE-2016-4236,CVE-2016-4237,CVE-2016-4238, CVE-2016-4239,CVE-2016-4240,CVE-2016-4241,CVE-2016-4242, CVE-2016-4243,CVE-2016-4244,CVE-2016-4245,CVE-2016-4246,CVE-2016-4182,CVE-2016-3375,CVE-2017-3001,CVE-2017-3002,CVE-2017-3003, CVE-2017-0238,CVE-2017-0236,CVE-2017-8549,CVE-2017-8619,CVE-2017-11887,CVE-2017-11913,CVE-2017-11846,CVE-2017-8753CVE-2018-8618,CVE-2018-8544,CVE-2018-8367,CVE-2018-8372, CVE-2018-8242,CVE-2018-8236,CVE-2018-1022,CVE-2018-0951,CVE-2018-0953,CVE-2018-8122,CVE-2018-8114,CVE-2018-0955,CVE-2018-0954,CVE-2018-1000,CVE-2018-0981,CVE-2018-0988,CVE-2018-0997, CVE-2018-1004,CVE-2018-0994,CVE-2018-0872,CVE-2018-0866,CVE-2018-0834,CVE-2018-0798,CVE-2018-0807,CVE-2018-0845,CVE-2018-0802,CVE-2018-0806,CVE-2018-0805,CVE-2018-0849,CVE-2018-0848, CVE-2018-0862,CVE-2018-0804,CVE-2018-0801,CVE-2018-0812,CVE-2019-1485,CVE-2019-1484,CVE-2019-1390,CVE-2019-1239,CVE-2019-1238,CVE-2019-1060,CVE-2019-1237,CVE-2019-1236,CVE-2019-1221, CVE-2019-1057,CVE-2019-1004,CVE-2019-1001,CVE-2019-1059,CVE-2019-1056,CVE-2019-0988,CVE-2019-1055,CVE-2019-1005,CVE-2019-0810,CVE-2019-0862,CVE-2019-0842,CVE-2019-0806,CVE-2019-0795,CVE-2019-0794,CVE-2019-0793,CVE-2019-0792,CVE-2019-0791,CVE-2019-0790,CVE-2019-0753,CVE-2019-0667,CVE-2019-0592,CVE-2019-0773,CVE-2019-0772,CVE-2019-0639,CVE-2019-0665,CVE-2019-0611,CVE-2019-0680,CVE-2019-0756,CVE-2019-0666,CVE-2019-0651,CVE-2019-0591,CVE-2019-0655,CVE-2019-0610,CVE-2019-0606,CVE-2019-0605,CVE-2019-0652,CVE-2019-0567,CVE-2020-16964,CVE-2020-16962,CVE-2020-16961,CVE-2020-16960,CVE-2020-16959,CVE-2020-16958,CVE-2020-16963,CVE-2020-17055,CVE-2020-17044,CVE-2020-17034,CVE-2020-17028,CVE-2020-17027,CVE-2020-17025,CVE-2020-17033,CVE-2020-17032,CVE-2020-17031,CVE-2020-17026,CVE-2020-16974,CVE-2020-16975,CVE-2020-16972,CVE-2020-16905,CVE-2020-16876,CVE-2020-16976,CVE-2020-16973,CVE-2020-16936,CVE-2020-16912,CVE-2020-16887,CVE-2020-1133,CVE-2020-1052,CVE-2020-0912,CVE-2020-1598,CVE-2020-1491,CVE-2020-1376,CVE-2020-1130,CVE-2020-0922,CVE-2020-1579,CVE-2020-1567,CVE-2020-1551,CVE-2020-1547,CVE-2020-1536,CVE-2020-1539,CVE-2020-1535,CVE-2020-1534,CVE-2020-1519,CVE-2020-1546,CVE-2020-1545,CVE-2020-1544,CVE-2020-1543,CVE-2020-1542,CVE-2020-1541,CVE-2020-1540,CVE-2020-1428,CVE-2020-1427,CVE-2020-1430,CVE-2020-1411,CVE-2020-1390,CVE-2020-1373,CVE-2020-1365,CVE-2020-1336,CVE-2020-1437,CVE-2020-1406,CVE-2020-1403,CVE-2020-1388,CVE-2020-1371,CVE-2020-1354,CVE-2020-1085,CVE-2020-1257,CVE-2020-1255,CVE-2020-1197,CVE-2020-1278,CVE-2020-1271,CVE-2020-1260,CVE-2020-1216,CVE-2020-1214,CVE-2020-1212,CVE-2020-1234,CVE-2020-1293,CVE-2020-1281,CVE-2020-1230,CVE-2020-1215,CVE-2020-1203,CVE-2020-1202,CVE-2020-1132,CVE-2020-1064,CVE-2020-1058,CVE-2020-1056,CVE-2020-1093,CVE-2020-1060,CVE-2020-1067,CVE-2020-1035,CVE-2020-1021,CVE-2020-1061,CVE-2020-1015,CVE-2020-0967,CVE-2020-0966,CVE-2020-0895,CVE-2020-0833,CVE-2020-0832,CVE-2020-0831,CVE-2020-0824,CVE-2020-0847,CVE-2020-0640,CVE-2021-40469,CVE-2021-40467,CVE-2021-40466,CVE-2021-26435,CVE-2021-36963,CVE-2021-34534,CVE-2021-33746,CVE-2021-34525,CVE-2021-34442,CVE-2021-34497,CVE-2021-34447,CVE-2021-34494,CVE-2021-33780,CVE-2021-33756,CVE-2021-33754,CVE-2021-33752,CVE-2021-33750,CVE-2021-33749,CVE-2021-31194,CVE-2021-28434,CVE-2021-28358,CVE-2021-28357,CVE-2021-28356,CVE-2021-28355,CVE-2021-28354,CVE-2021-28353,CVE-2021-28352,CVE-2021-28346,CVE-2021-28345,CVE-2021-28344,CVE-2021-28343,CVE-2021-28342,CVE-2021-28341,CVE-2021-28340,CVE-2021-28339,CVE-2021-28338,CVE-2021-28337,CVE-2021-28336,CVE-2021-28335,CVE-2021-28334,CVE-2021-28333,CVE-2021-28332,CVE-2021-28331,CVE-2021-28330,CVE-2021-28329,CVE-2021-27088,CVE-2021-26901,CVE-2021-26899,CVE-2021-26898,CVE-2021-26872,CVE-2021-24103,CVE-2021-24102,CVE-2021-1673,CVE-2021-1667,CVE-2021-1658,CVE-2021-1649,CVE-2021-1702,CVE-2021-1701,CVE-2021-1700,CVE-2021-1671,CVE-2021-1666,CVE-2021-1664,CVE-2021-1660,CVE-2022-44670,CVE-2022-44676,CVE-2022-41039,CVE-2022-41088,CVE-2022-41081,CVE-2022-30198,CVE-2022-22035,CVE-2022-33634,CVE-2022-24504,CVE-2022-38000,CVE-2022-38048,CVE-2022-38047,CVE-2022-35830,CVE-2022-34722,CVE-2022-34721,CVE-2022-35794,CVE-2022-35767,CVE-2022-35766,CVE-2022-35753,CVE-2022-35752,CVE-2022-35745,CVE-2022-35744,CVE-2022-34714,CVE-2022-34702,CVE-2022-22039,CVE-2022-22038,CVE-2022-22029,CVE-2022-30161,CVE-2022-30153,CVE-2022-30149,CVE-2022-30146,CVE-2022-30143,CVE-2022-30142,CVE-2022-30141,CVE-2022-30139,CVE-2022-30136,CVE-2022-29141,CVE-2022-29139,CVE-2022-29137,CVE-2022-29131,CVE-2022-29130,CVE-2022-29129,CVE-2022-29128,CVE-2022-22014,CVE-2022-22013,CVE-2022-22012,CVE-2022-26937,CVE-2022-21972,CVE-2022-23270,CVE-2022-26919,CVE-2022-26825,CVE-2022-26824,CVE-2022-26823,CVE-2022-26822,CVE-2022-26821,CVE-2022-26820,CVE-2022-26819,CVE-2022-26818,CVE-2022-26817,CVE-2022-26815,CVE-2022-26814,CVE-2022-26813,CVE-2022-26812,CVE-2022-26811,CVE-2022-24500,CVE-2022-24497,CVE-2022-24541,CVE-2022-24492,CVE-2022-24534,CVE-2022-24485,CVE-2022-24528,CVE-2022-21983,CVE-2023-24903,CVE-2023-28283,CVE-2023-28240,CVE-2023-28238,CVE-2023-28220,CVE-2023-28219,CVE-2023-23404,CVE-2023-23414,CVE-2023-23407,CVE-2023-23385,CVE-2023-21692,CVE-2023-21712,CVE-2023-21679,CVE-2023-21556,CVE-2023-21555,CVE-2023-21548,CVE-2023-21546,CVE-2023-21535

Highlights of This Session

A walk through of a bug hunting project

✓ Windows RAS VPN components

✓ Examples of pre-auth remote bugs & bug patterns in windows RAS

✓ Not only result but also approach & thoughts during the research

✓ Windows bounty experience

✕ Exploiting details of the bugs is beyond the scope

Agenda

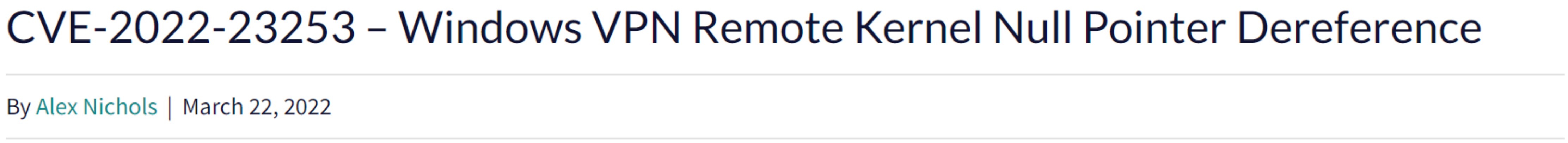
- Background
- Windows Remote Access Service
- PPTP
- Authentication Protocols
- SSTP
- L2TP
- IKE
- Future Work & Take Aways

black hat USA 2023

Background

Initiative of the Research

- Read this blog last April



CVE-2022-23253 – Windows VPN Remote Kernel Null Pointer Dereference

By Alex Nichols | March 22, 2022

- Alex Nicols (@i4mchr00t) blogs a remote Null Pointer DoS bug he found in Windows PPTP
- Root cause: Failed to handle the case where control commands are sent in wrong sequence

Why it’s Interesting – From a Bug Bounty Hunter’s View

- No Windows PPTP server bugs discussed before
  - – Blue Ocean
  - - The bug is fresh: not many competitors now
- The bug looks relatively simple
  - – Code quality not so good & Not well audited
- Remote & Pre-auth & No user interaction & Server side

Introduction to Microsoft WIP Bounty Program

General Awards

Security Impact	Maximum Award
Remote Code Execution	$5,000
Elevation of Privilege	$2,000
Security Feature Bypass	$1,000
Information Disclosure	$1,000
Spoofing	$1,000
Tampering	$1,000
Denial of Service	$500

Attack Scenario Awards*

Attack Vector	Scenario	Maximum Award
Remote (assumes no prior execution)	Unauthenticated[1] non-sandboxed code execution with no user interaction	$100,000
	Demonstrated[2] unauthenticated and unauthorized access to private[3] user data or data that can be used to weaken existing user protections with little[4] or no user interaction	$50,000
	Unauthenticated data destruction or persistent denial of service with no user interaction	$30,000
Local (assumes prior execution)	Sandbox[5] escape with little or no user interaction	$20,000
	Demonstrated unauthorized access to private user data from a sandboxed[5] process with no user interaction	$20,000

Attack Scenario – Before Starting

- High risk high return: more difficult to find bug than general bugs
- Before starting

  - Read every line in the bounty page carefully

  - Check the up-to-date out-of-scope section timely

  - Fully understand the attack scenarios

    * Web browser RCE User Interaction

    * Office RCE when user opens a document User Interaction

    * Domain user achieved RCE in domain controller Authentication

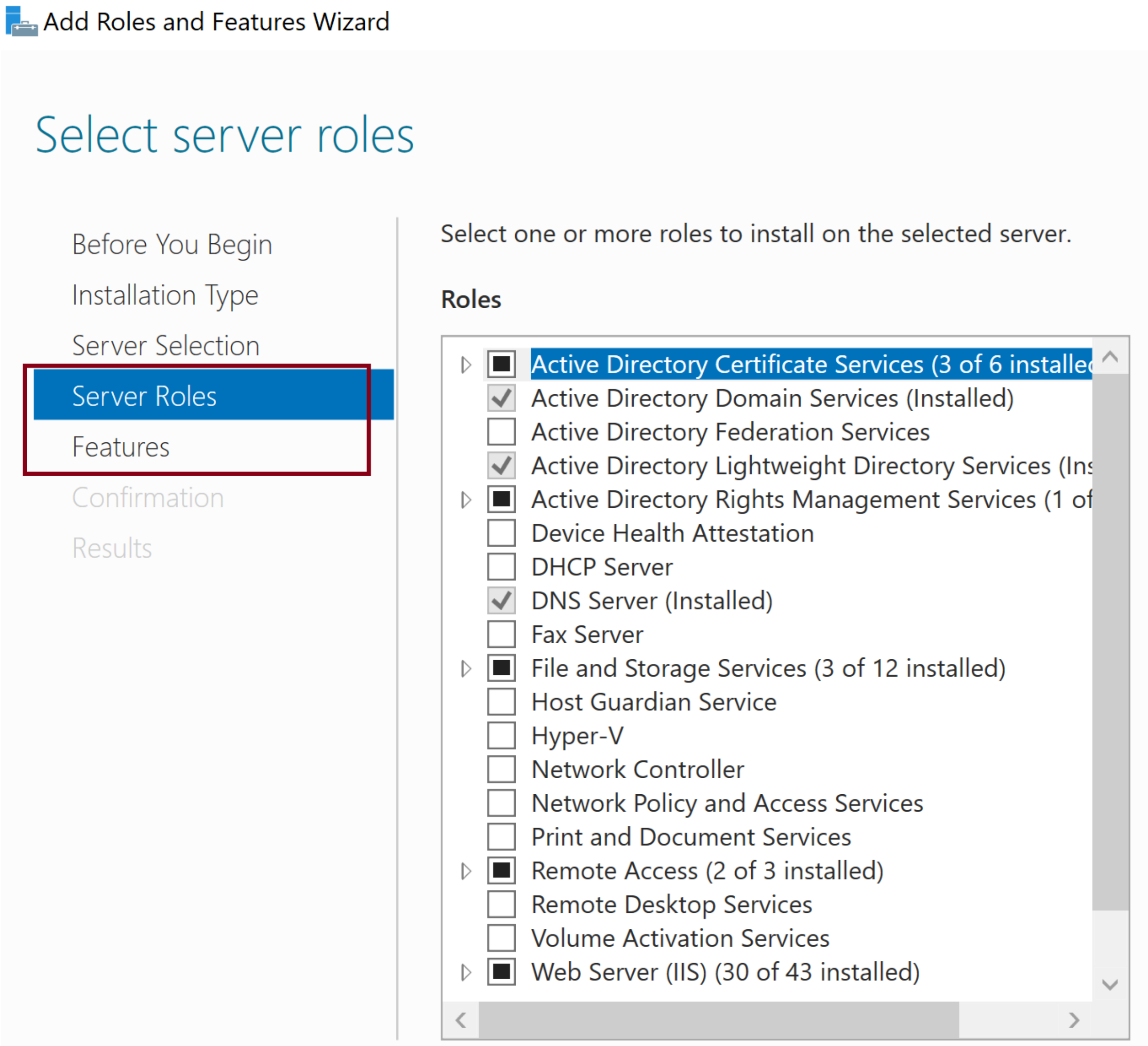
    * Pre-auth, no user interaction, but non-default configuration **Depends on reviewer**

Where are the Pre-auth Targets

- Learn & get inspired from other researchers' work

  - Researchers on MSRC MVP list: their recent bugs?

- Explore & learn Windows features for you potential pre-auth target



Add Roles and Features Wizard

Select server roles

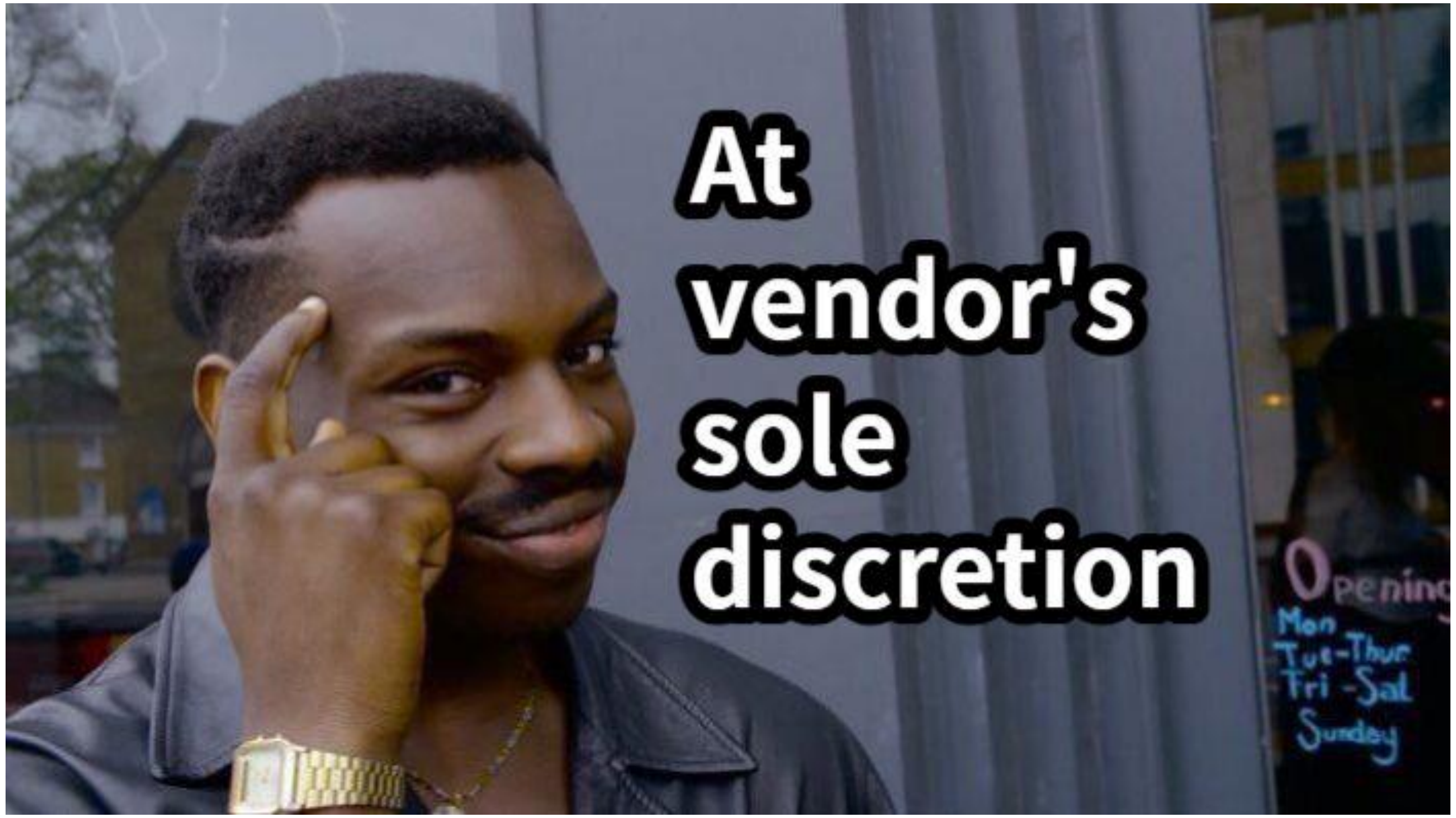
Select one or more roles to install on the selected server.

Roles

- ■ Active Directory Certificate Services (3 of 6 installe
- ☑ Active Directory Domain Services (Installed)
- □ Active Directory Federation Services
- ☑ Active Directory Lightweight Directory Services (Ins
- ■ Active Directory Rights Management Services (1 of
- □ Device Health Attestation
- □ DHCP Server
- ☑ DNS Server (Installed)
- □ Fax Server
- ■ File and Storage Services (3 of 12 installed)
- □ Host Guardian Service
- □ Hyper-V
- □ Network Controller
- □ Network Policy and Access Services
- □ Print and Document Services
- ■ Remote Access (2 of 3 installed)
- □ Remote Desktop Services
- □ Volume Activation Services
- ■ Web Server (IIS) (30 of 43 installed)

A Good Temper is the Most Important When Playing Bug Bounty Program

- Repeat below sentence 1000 times before you starting

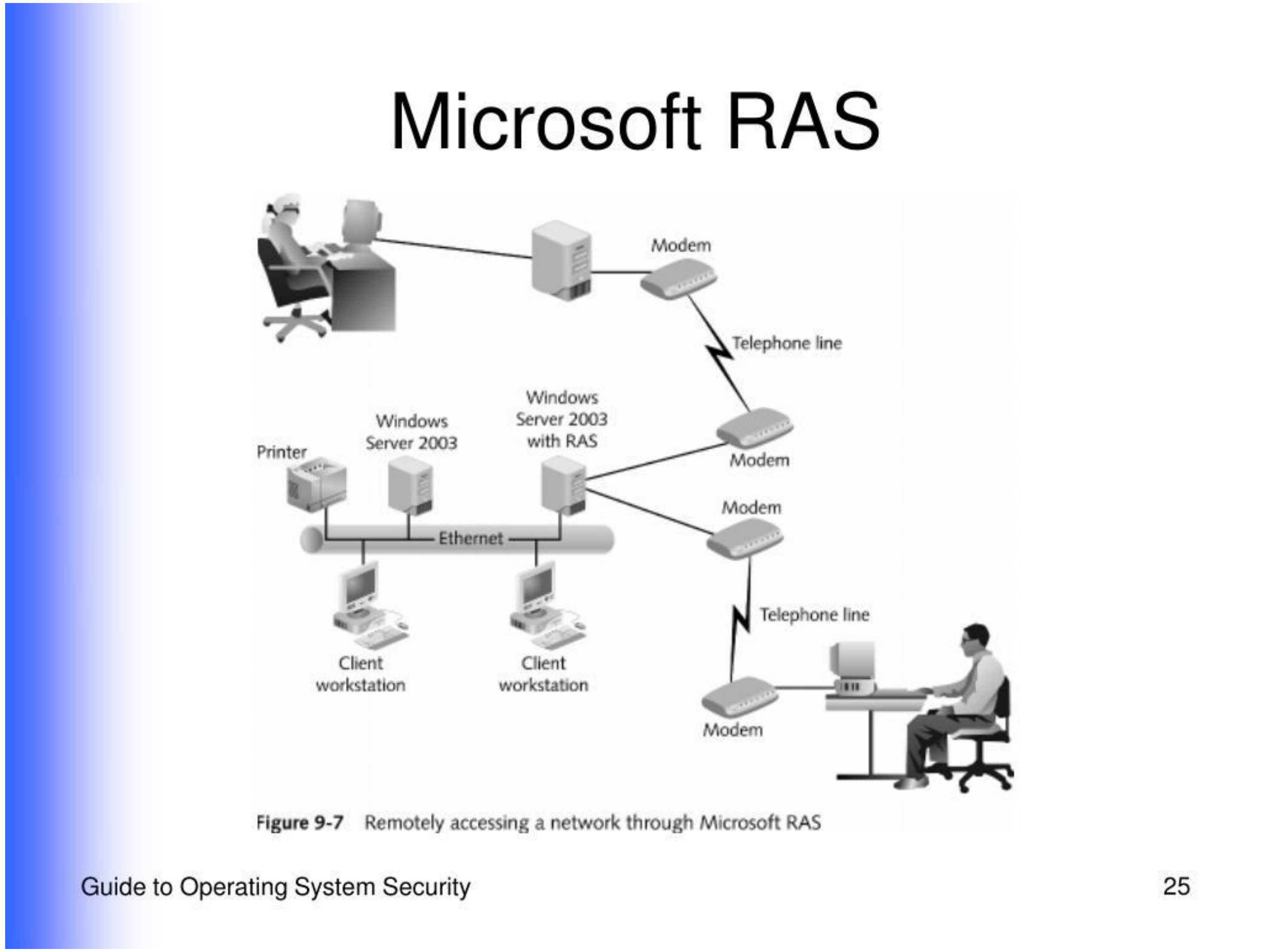


black hat USA 2023

Windows Remote Access Service

Windows Remote Access Service

- Provides remote access to clients
- **Direct Access & VPN**
- Routing
- Proxy
- Implemented in numbers of kernel drivers & user mode services
- Windows Server & Azure Cloud



Microsoft RAS

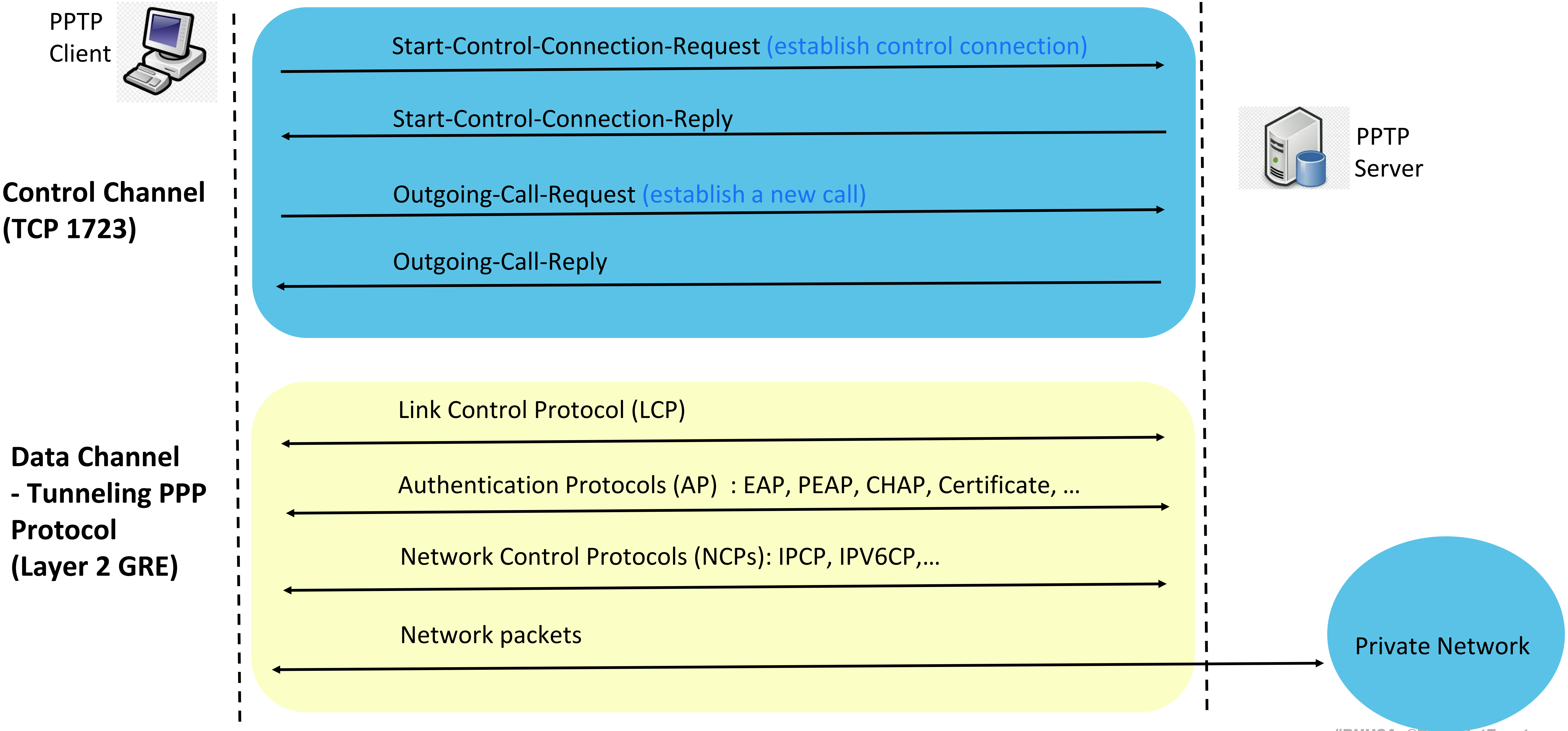
Figure 9-7 Remotely accessing a network through Microsoft RAS

Guide to Operating System Security 25

PPTP

PPTP - Point-to-Point Tunneling Protocol

- A decades-old protocol for VPN
- Use 2 channels
- Control channel – TCP 1723
- Data channel – Use Generic Routing Encapsulation (GRE) to encapsulate the PPP (Point-to-Point Protocol)



PPTP Client

PPTP Server

Control Channel (TCP 1723)

Start-Control-Connection-Request (establish control connection)

Start-Control-Connection-Reply

Outgoing-Call-Request (establish a new call)

Outgoing-Call-Reply

Data Channel - Tunneling PPP Protocol (Layer 2 GRE)

Link Control Protocol (LCP)

Authentication Protocols (AP) : EAP, PEAP, CHAP, Certificate, …

Network Control Protocols (NCPs): IPCP, IPV6CP,…

Network packets

Private Network

Components & Pre-auth Attack Surfaces

Main control channel handling, Data channel kernel dispatch:
raspptp.sys

Framework to manage calls, used by raspptp.sys:
ndis.sys

Control Channel

PPP Tunneling: **raspptp.sys**
ndis.sys ndiswan.sys

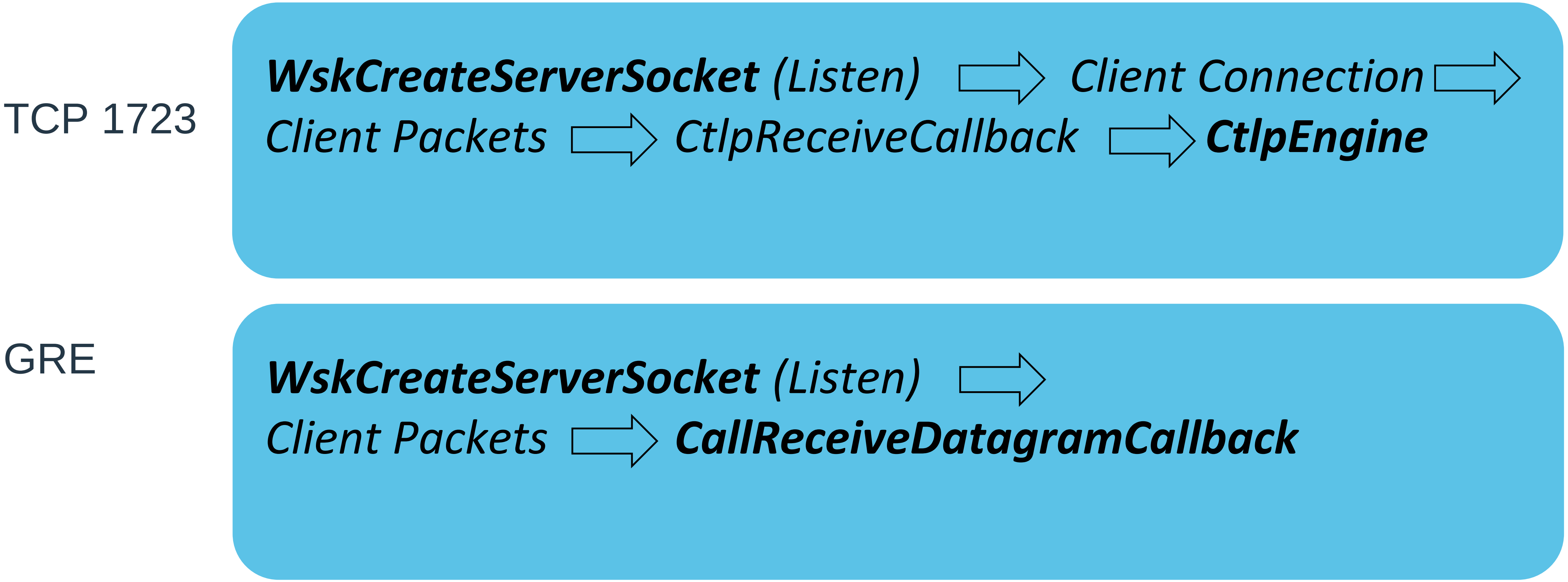
PPP engine: **rasppp.dll**

Authentication protocols:
raseap.dll, rastls.dll,
raschap.dll, ……

Data Channel

Raspptp.sys

- TCP port 1723 & GRE packets
- Matalins control channel & call states
- The first and most straightforward component to look for bugs



TCP 1723

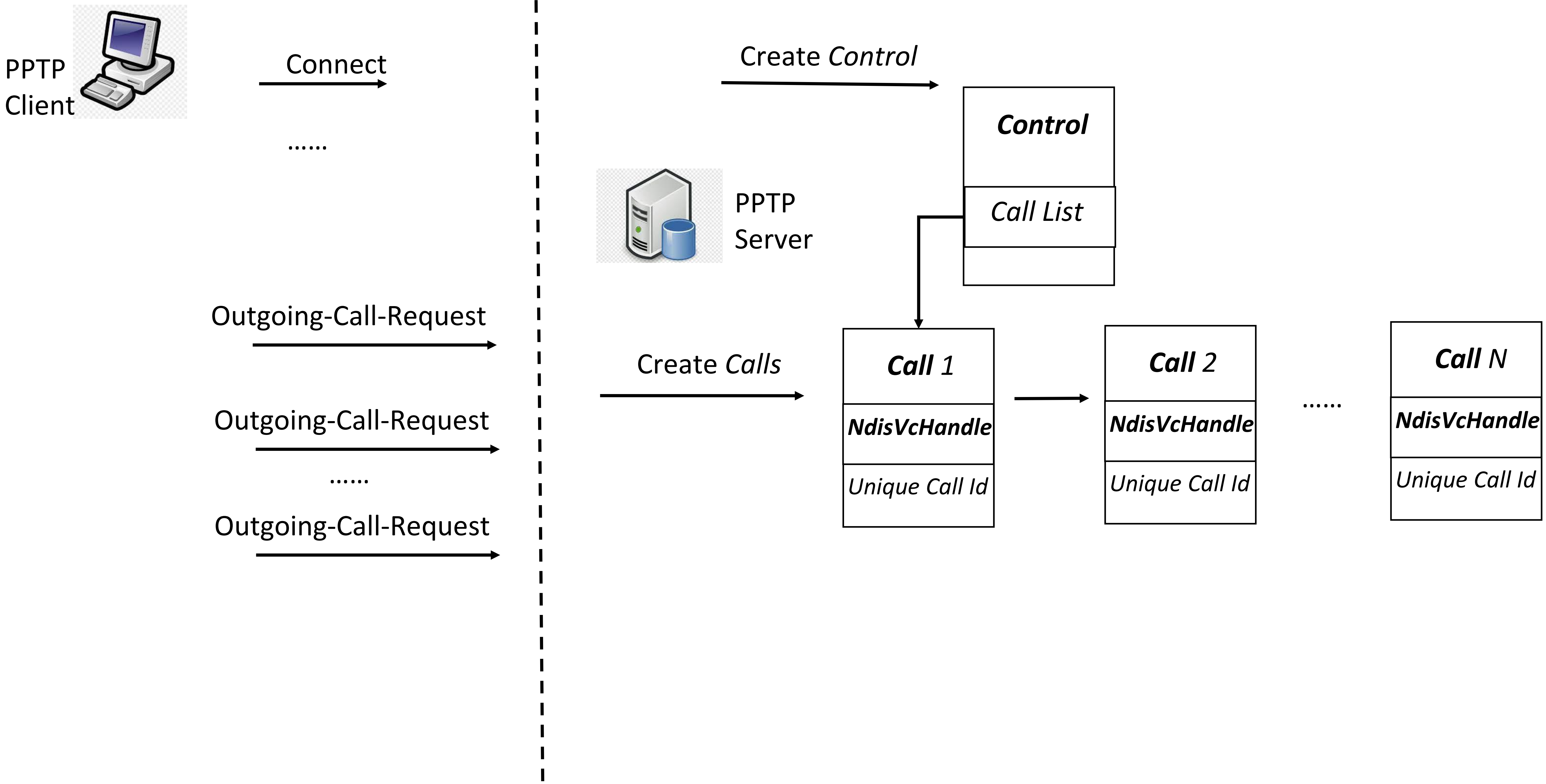
WskCreateServerSocket *(Listen)* → *Client Connection* → *Client Packets* → *CtlpReceiveCallback* → ***CtlpEngine***

GRE

WskCreateServerSocket *(Listen)* → *Client Packets* → ***CallReceiveDatagramCallback***

Raspptp.sys - Key Data Structures

- ***Control***: Represents a control channel between client and server, a *ControlObject* is created when a client successfully connects to the server TCP port 1723
- ***Call***: Represents a PPTP call between client and server, a *CallObject* is created when the server handles clients' Outgoing-Call-Request/Incoming-Call-Request. Multiple *Calls* can be associated to a single *Control*, and are linked in a double-linked list
- ***NdisVcHandle***: A NDIS_HANDLE, represents a virtual call in the NDIS framework, used by both kernel driver and user mode service to access a PPTP connection. Each *Call contains a NdisVcHandle*



PPTP Client

Connect

……

Outgoing-Call-Request

Outgoing-Call-Request

……

Outgoing-Call-Request

PPTP Server

Create *Control*

Control

Call List

Create *Calls*

***Call** 1*

NdisVcHandle

Unique Call Id

***Call** 2*

NdisVcHandle

Unique Call Id

……

***Call** N*

NdisVcHandle

Unique Call Id

Raspptp.sys - Where to Look for Bugs

- Packet/Message parsing/handling

  - Control messages/State machine

  - PPP framing (embedded in GRE)

- Object life cycle management

  - Control, Call, NdisVcHandle, with complex cross references

  - Multi-threading access

  - **Use after free/Race condition**

Race Condition – What the Developer Think

Race Condition – What's Actual Happening

A Simple Fuzzer for Quick Proofing

- Developed a simple fuzzer:

  1. Create a connection to server

  2. Create some calls for this connection

  3. Create some threads which randomly perform below actions:

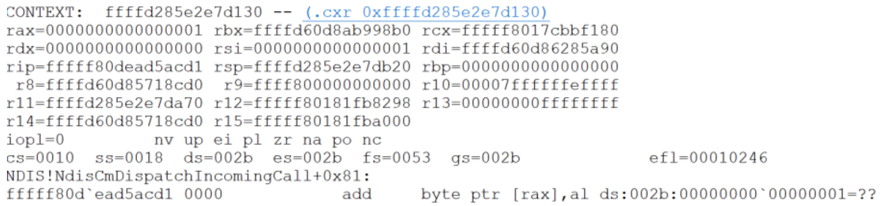
     2.1 Send call related messages to the server (create/destroy/setting)

     2.2 Send control related messages to the server (create/destroy)

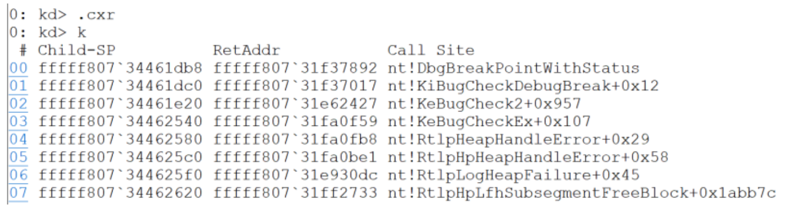
     2.3 Close the connection

Result of the Fuzzing

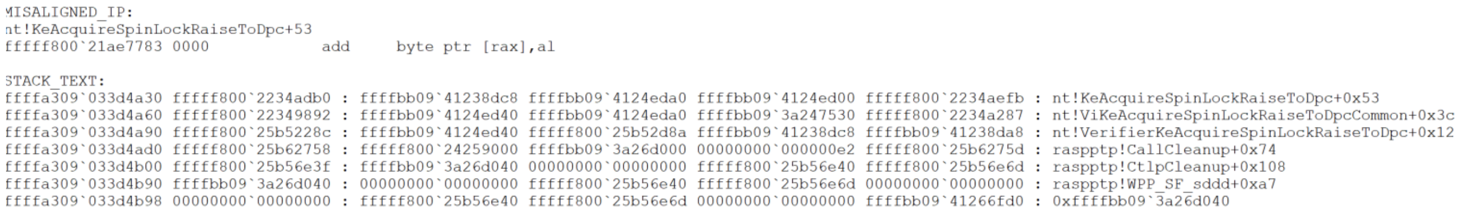
- Got multiple UAF/NPD race condition crashes in minutes



```
CONTEXT:  ffffd285e2e7d130 -- (.cxr 0xffffd285e2e7d130)
rax=0000000000000001 rbx=ffffd60d8ab998b0 rcx=fffff8017cbbf180
rdx=0000000000000000 rsi=0000000000000001 rdi=ffffd60d86285a90
rip=fffff80dead5acd1 rsp=ffffd285e2e7db20 rbp=0000000000000000
 r8=ffffd60d85718cd0  r9=ffff800000000000 r10=00007ffffffeffff
r11=ffffd285e2e7da70 r12=fffff80181fb8298 r13=00000000ffffffff
r14=ffffd60d85718cd0 r15=fffff80181fba000
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
NDIS!NdisCmDispatchIncomingCall+0x81:
fffff80d`ead5acd1 0000            add     byte ptr [rax],al ds:002b:00000000`00000001=??
```



```
0: kd> .cxr
0: kd> k
 # Child-SP          RetAddr           Call Site
00 fffff807`34461db8 fffff807`31f37892 nt!DbgBreakPointWithStatus
01 fffff807`34461dc0 fffff807`31f37017 nt!KiBugCheckDebugBreak+0x12
02 fffff807`34461e20 fffff807`31e62427 nt!KeBugCheck2+0x957
03 fffff807`34462540 fffff807`31fa0f59 nt!KeBugCheckEx+0x107
04 fffff807`34462580 fffff807`31fa0fb8 nt!RtlpHeapHandleError+0x29
05 fffff807`344625c0 fffff807`31fa0be1 nt!RtlpHpHeapHandleError+0x58
06 fffff807`344625f0 fffff807`31e930dc nt!RtlpLogHeapFailure+0x45
07 fffff807`34462620 fffff807`31ff2733 nt!RtlpHpLfhSubsegmentFreeBlock+0x1abb7c
```



```
MISALIGNED_IP:
nt!KeAcquireSpinLockRaiseToDpc+53
fffff800`21ae7783 0000            add     byte ptr [rax],al

STACK_TEXT:
ffffa309`033d4a30 fffff800`2234adb0 : ffffbb09`41238dc8 ffffbb09`4124eda0 ffffbb09`4124ed00 fffff800`2234aefb : nt!KeAcquireSpinLockRaiseToDpc+0x53
ffffa309`033d4a60 fffff800`22349892 : ffffbb09`4124ed40 ffffbb09`4124eda0 ffffbb09`3a247530 fffff800`2234a287 : nt!ViKeAcquireSpinLockRaiseToDpcCommon+0x3c
ffffa309`033d4a90 fffff800`25b5228c : ffffbb09`4124ed40 fffff800`25b52d8a ffffbb09`41238dc8 ffffbb09`41238da8 : nt!VerifierKeAcquireSpinLockRaiseToDpc+0x12
ffffa309`033d4ad0 fffff800`25b62758 : fffff800`24259000 ffffbb09`3a26d000 00000000`000000e2 fffff800`25b6275d : raspptp!CallCleanup+0x74
ffffa309`033d4b00 fffff800`25b56e3f : ffffbb09`3a26d040 00000000`00000000 fffff800`25b56e40 fffff800`25b56e6d : raspptp!CtlpCleanup+0x108
ffffa309`033d4b90 ffffbb09`3a26d040 : 00000000`00000000 fffff800`25b56e40 fffff800`25b56e6d 00000000`00000000 : raspptp!WPP_SF_sddd+0xa7
ffffa309`033d4b98 00000000`00000000 : fffff800`25b56e40 fffff800`25b56e6d 00000000`00000000 ffffbb09`41266fd0 : 0xffffbb09`3a26d040
```

What’s Next

- The fuzzing result gives a quick proof

  - raspptp.sys is vulnerable to race condition

- Time for manual auditing

  Found 20+ race condition RCEs by manual auditing

Case Study – Call Use After Free

Thread 1 – Client sends Call-Disconnect-Notify request with a CallId

```
CtlpEngine
{
…
For each Call in Control.CallList:
    if Call.id == CallId:
      break
// No Lock, no reference counter
CallEventCallDisconnectNotify(Call)
}
```

Race Window

Thread 2 – Client close the same connection

```
CtlpCleanup
{
…
For each Call in Control.CallList:
 // Free the call, no lock
      CallCleanup(Call)
}
```

Case Study – Control Use After Free

Thread 1 – Client closes the connection

⇩

```
CtlDisconnectCallback
{
// No Lock, no reference counter
 CtlSetState(pControl, 7)
}
```

Race Window

Thread 2 – Client sends a Stop-Control-Connection-Request

⇩

```
CtlpCleanup
{
// Free the control object
CtrFree(pControl)
}
```

Case Study – NdisVcHandle After Free

Thread 1 – Client sends Incoming-Call-Request

CallpNewIncomingConnection

{

// No Lock, no reference counter

NdisCmDispatchIncomingCall (...,

pCall->NdisVcHandle

,...)

}

Race Window

Thread 2 – Client sends a Call-Disconnect-Notify

CallCleanup

{

// Free the NdisVcHandle

NdisMCmDeleteVc(pCall->NdisVcHandle);

}

Case Study – Call object race UAF between data Channel and control channel

Thread 1 – Client sends a GRE packet to the server (data channel)

```
CallReceiveDatagramCallback
{
pCall = CallGetCall(callid_from_gre_pkt)
// No Lock, no reference counter
Use pCall
}
```

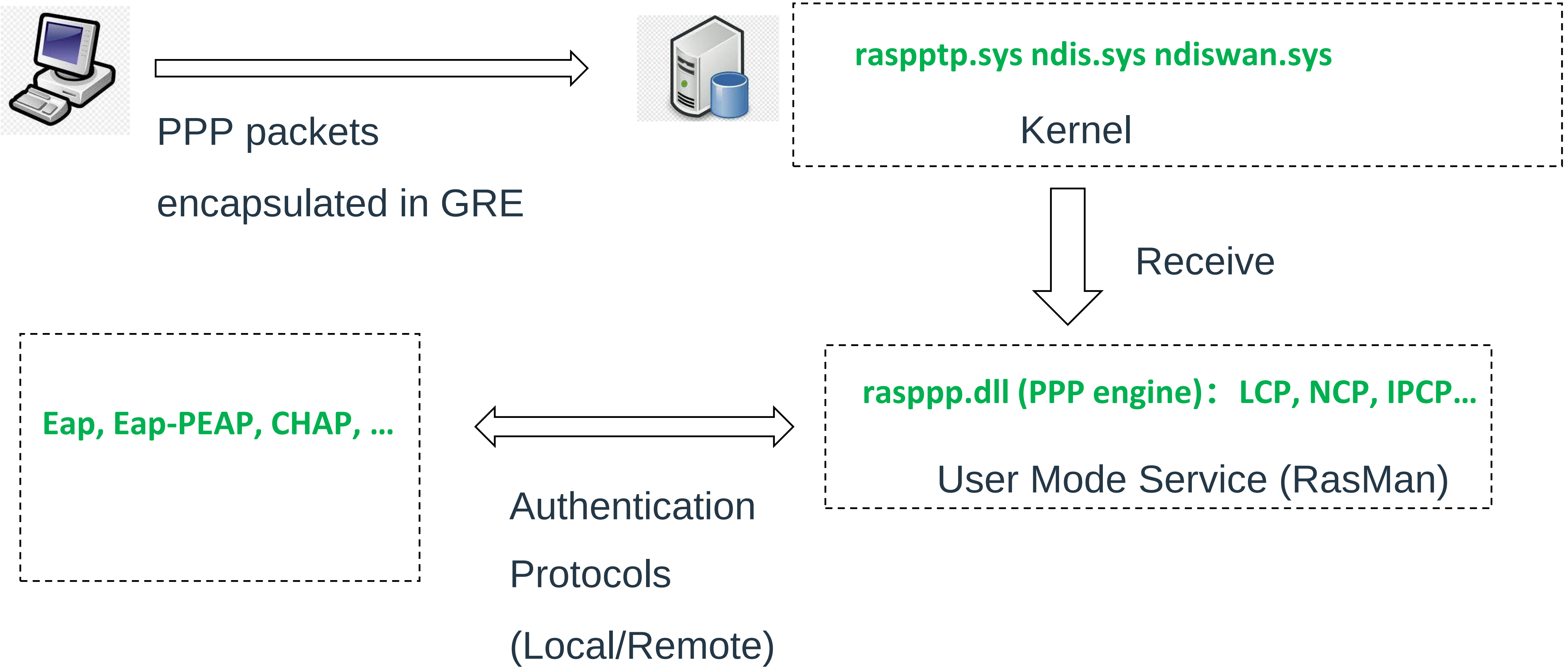
Race Window

Thread 2 – Client close the TCP control channel connection (control channel)

```
CtlpCleanup
{
…
For each Call in Control.CallList:
 // Free the call, no lock
      CallCleanup(pCall)
}
```

User Mode Service

- Rasppp.dll: User mode handler of the PPP protocol



PPP packets

encapsulated in GRE

raspptp.sys ndis.sys ndiswan.sys

Kernel

Receive

rasppp.dll (PPP engine)： LCP, NCP, IPCP…

User Mode Service (RasMan)

Eap, Eap-PEAP, CHAP, …

Authentication

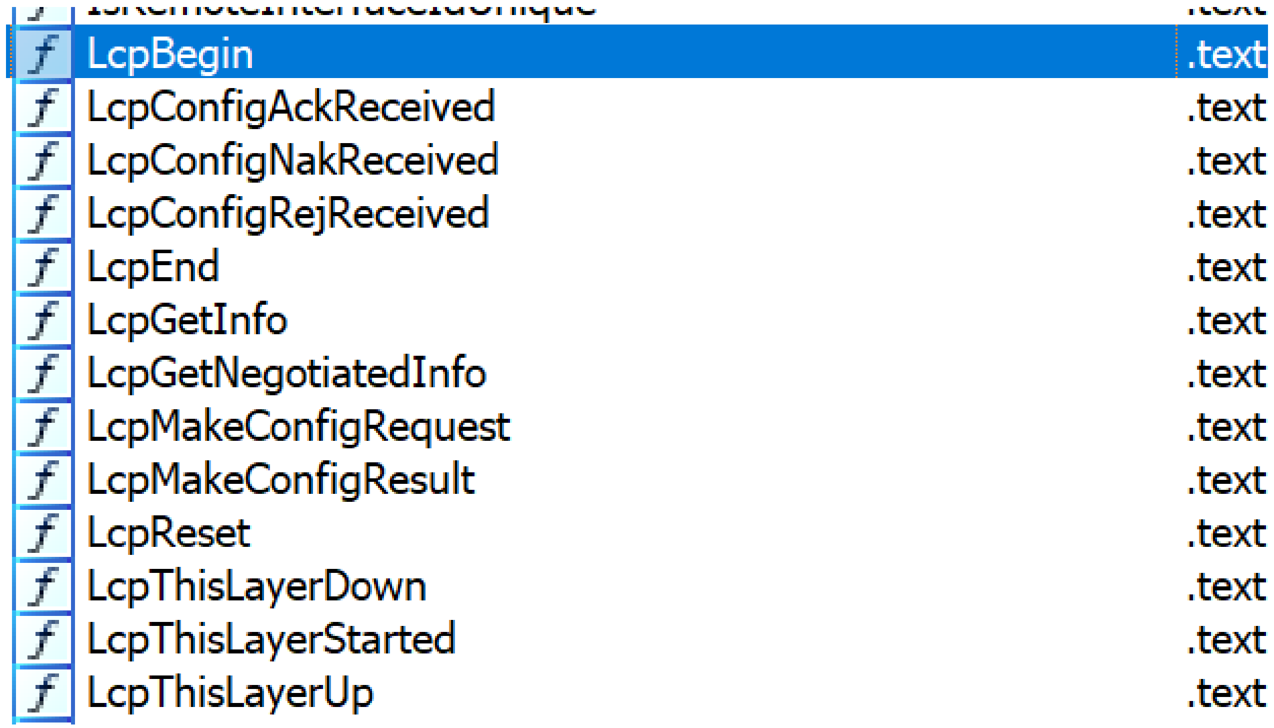
Protocols

(Local/Remote)

LCP - Link Control Protocol

- Establishing, configuring, testing, maintaining, negating options and terminating links for transmission

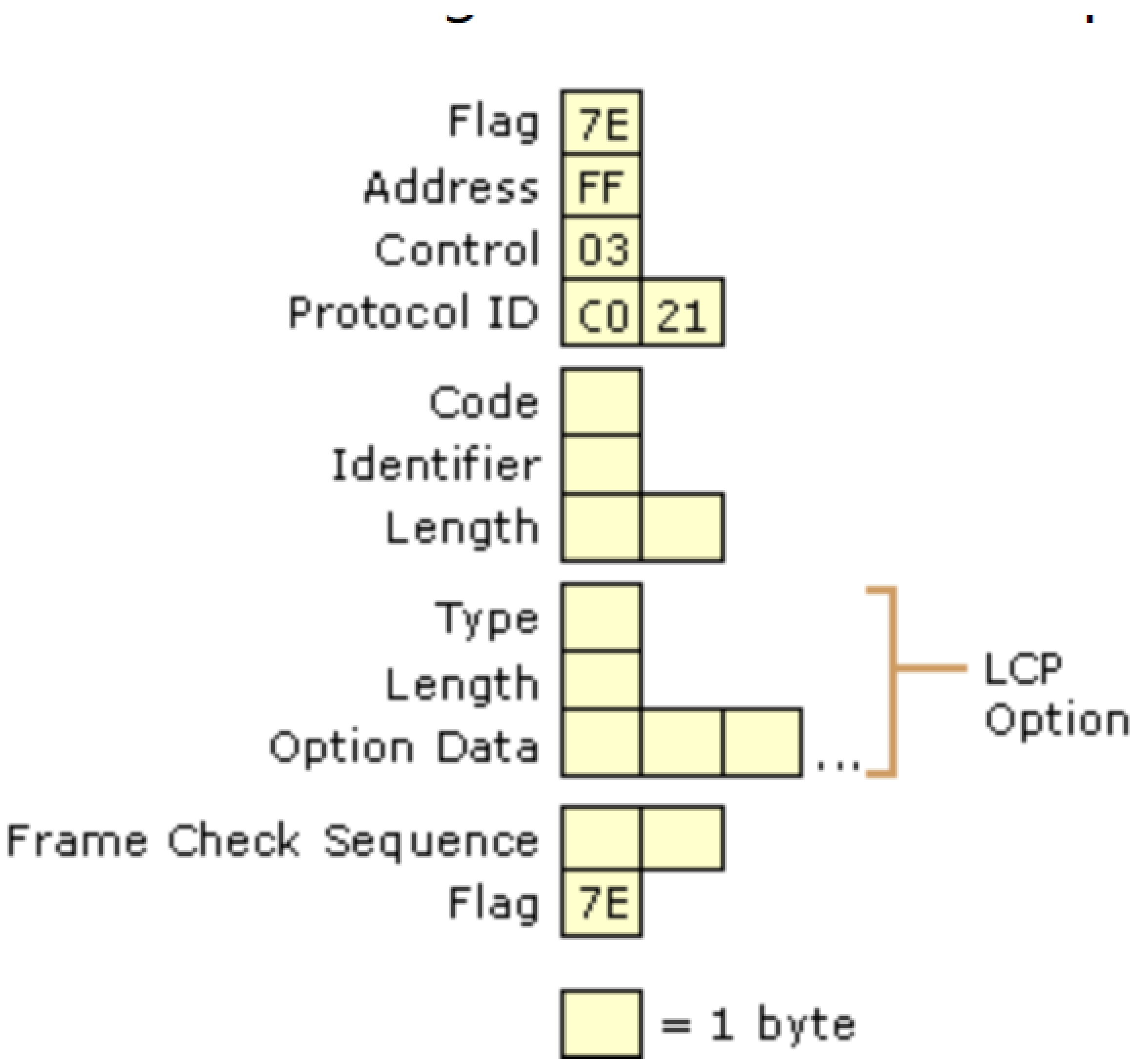
Pre-auth



```
IsRemoteInterfaceIdUnique    .text
LcpBegin                     .text
LcpConfigAckReceived         .text
LcpConfigNakReceived         .text
LcpConfigRejReceived         .text
LcpEnd                       .text
LcpGetInfo                   .text
LcpGetNegotiatedInfo         .text
LcpMakeConfigRequest         .text
LcpMakeConfigResult          .text
LcpReset                     .text
LcpThisLayerDown             .text
LcpThisLayerStarted          .text
LcpThisLayerUp               .text
```

Case Study - LcpMakeConfigResult Heap OOB Read/Write

- Client send a Configure-Request to the server
- Server calls LcpMakeConfigResult, which will parse the request packet
- The request packet data contains 0 ~ N LCP options



Spot the Problem?

```
DWORD LcpMakeConfigResult(…)
{
        //
        // Both src and dst options buffer size is same
        //
        PPP_OPTION* pSrcOption; // From remote client
        PPP_OPTION* pDstOption; // The output options

        while (lSrcLength > 0) {
                memcpy(pDstOption, pSrcOption, pSrcOption->length);
                pSrcOption += pSrcOption->length;

                pDstOption += pDstOption->length;

                lSrcLength -= pSrcOption->length;
        }
}
```

```
LCP_OPTION
{
BYTE type;
BYTE length;
Byte Data[];
}
```

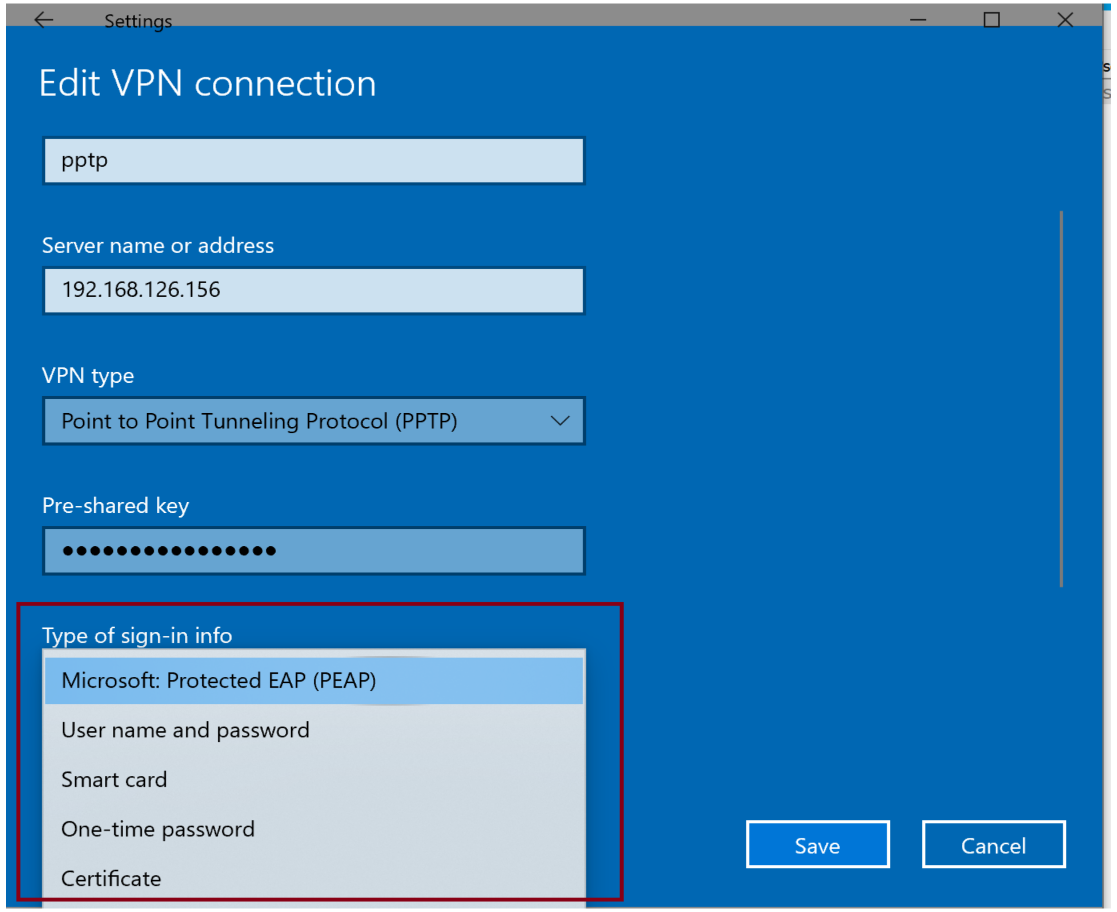
If pSrcOption->length < 2,
then pDstOption->length will be uninitialized data,
after pDstOption += pDstOption->length,
pDstOption can exceed the buffer end

black hat USA 2023

Authentication Protocols

Authentication Protocol (AP)

- The PPP engine will call local/remote authentication service to do the real authentication
- Windows RAS supports different auth methods
- If any AP algorithm contains a bug before auth finish, it’s also pre-auth



Edit VPN connection

pptp

Server name or address

192.168.126.156

VPN type

Point to Point Tunneling Protocol (PPTP)

Pre-shared key

●●●●●●●●●●●●●●●●

Type of sign-in info

Windows VPN Authentication Protocols

- Extensible Authentication Protocol (EAP) for built-in VPN types (IKEv2, L2TP, PPTP, SSTP)
- EAP-MSCHAPv2

  - username/password, windows logon credential
- EAP-TLS

  - certificate authentication
- EAP-PEAP

  - server validation + EAP-MSCHAPv2/EAP-TLS
- TTLS

  - PAP/CHAP/MSCHAP/MSCHAPV2

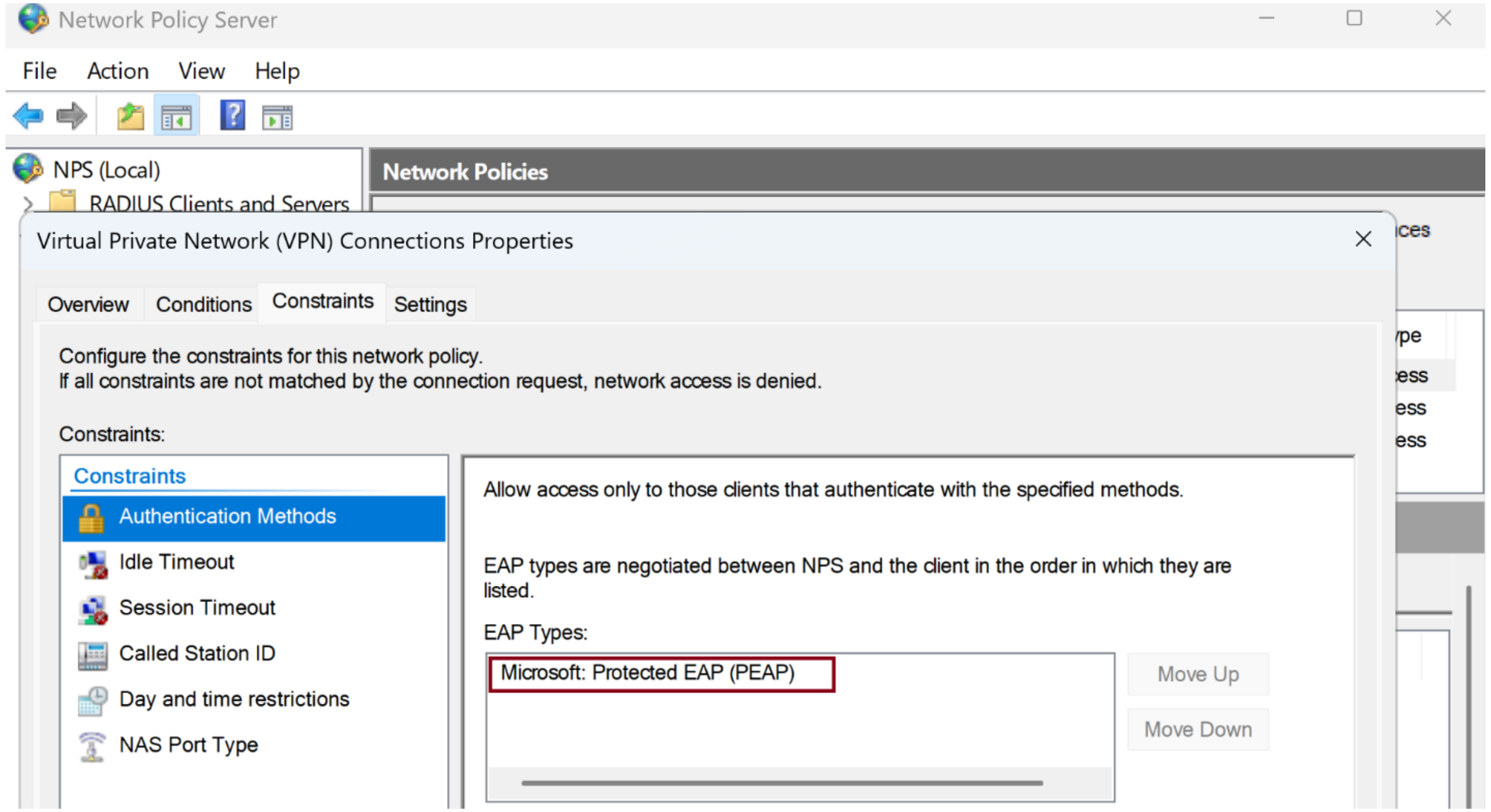
Case Study – PEAP Buffer OOB Write

- Microsoft protected EAP

  EAP authentication with TLS support

  More secure

  Implemented in eaptls.dll



Network Policy Server

Virtual Private Network (VPN) Connections Properties

Configure the constraints for this network policy.
If all constraints are not matched by the connection request, network access is denied.

Constraints:

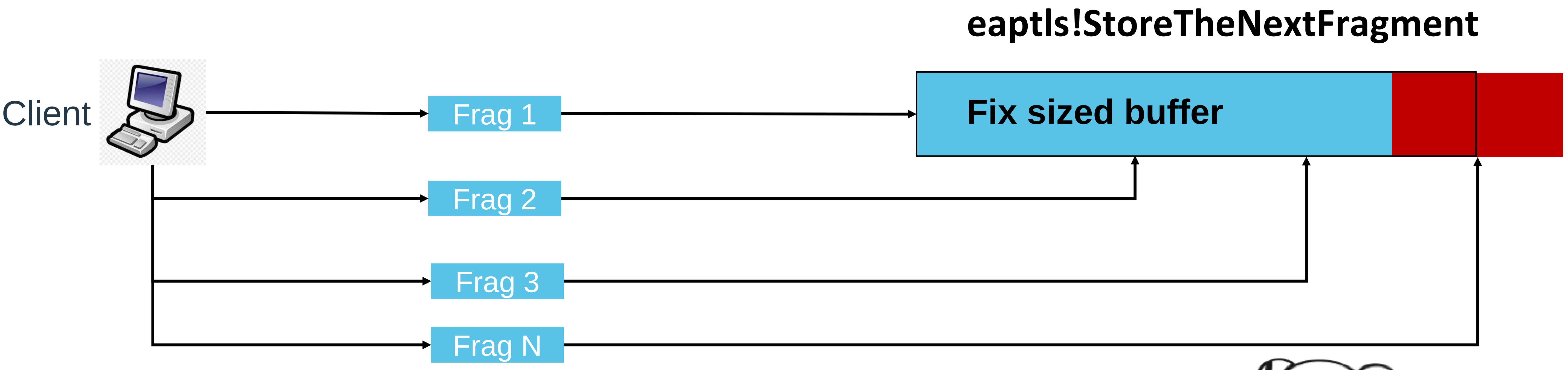
Allow access only to those clients that authenticate with the specified methods.

EAP types are negotiated between NPS and the client in the order in which they are listed.

EAP Types:

Microsoft: Protected EAP (PEAP)

PEAP – Fragmentation and Reassembling

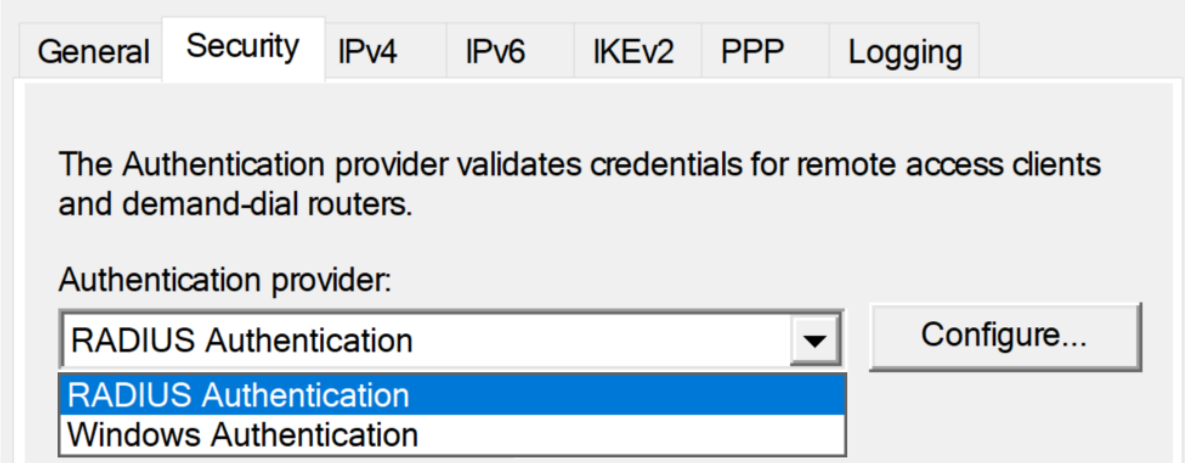


No boundary check

Useful tip: Whenever you see a protocol that supports fragmentation, be sure to check the fragmentation implementation.

RADIUS and Windows NPS

- RADIUS: a client-server protocol for authentication and accounting
- Windows VPN server can be configured to use RADIUS authentication
- NPS (Network Policy Server) is the Microsoft implementation of the RADIUS standard



The Authentication provider validates credentials for remote access clients and demand-dial routers.

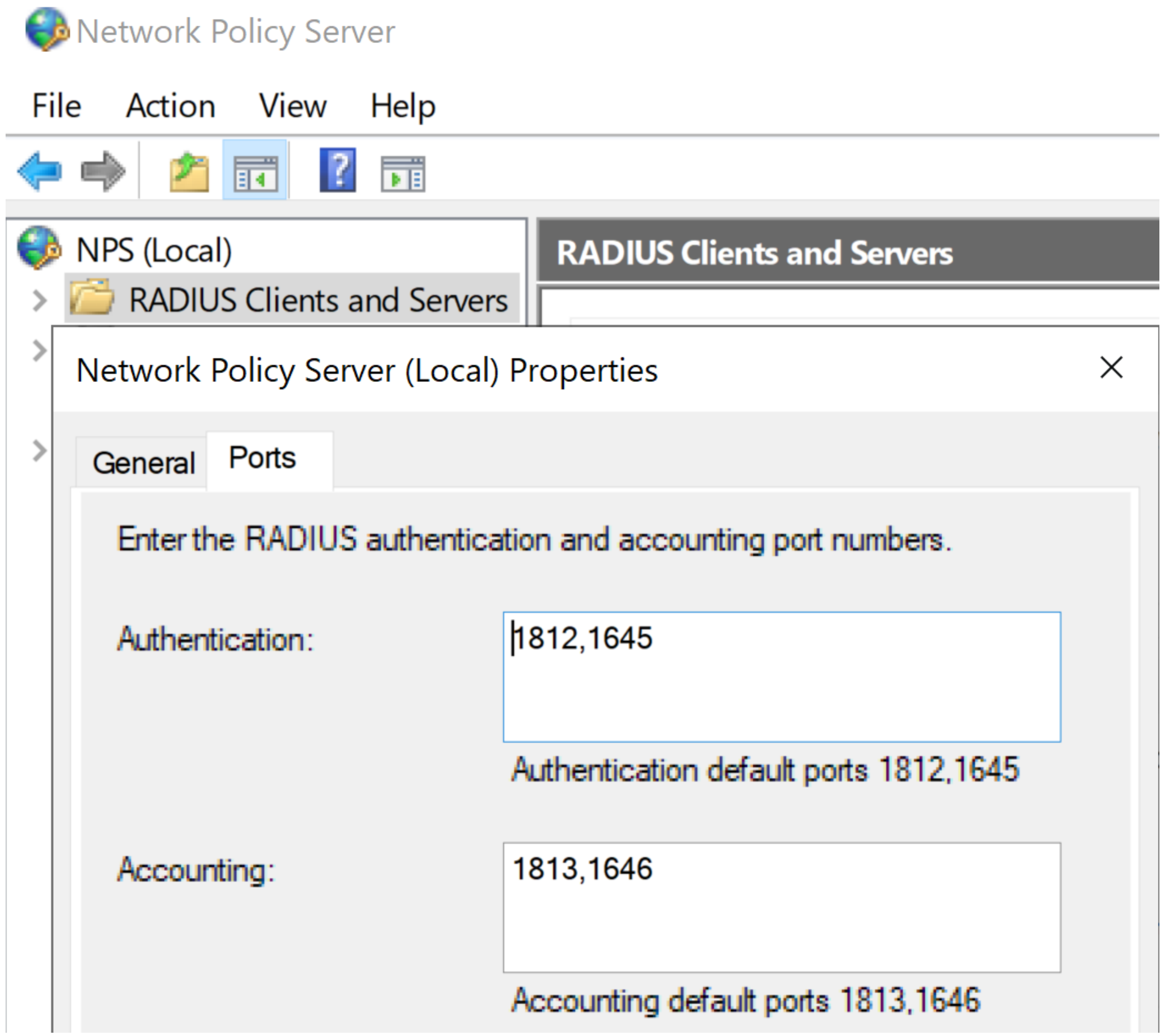
Authentication provider:

RADIUS Authentication

RADIUS Authentication
Windows Authentication

Windows NPS – Pre-auth Attack Surface

- iasrad.dll
- UDP ports (1812/1813, 1645/1646)
- Focus on RADIUS packets handling before authentication finish



Network Policy Server
File
Action
View
Help
NPS (Local)
RADIUS Clients and Servers
RADIUS Clients and Servers
Network Policy Server (Local) Properties
General
Ports

Enter the RADIUS authentication and accounting port numbers.

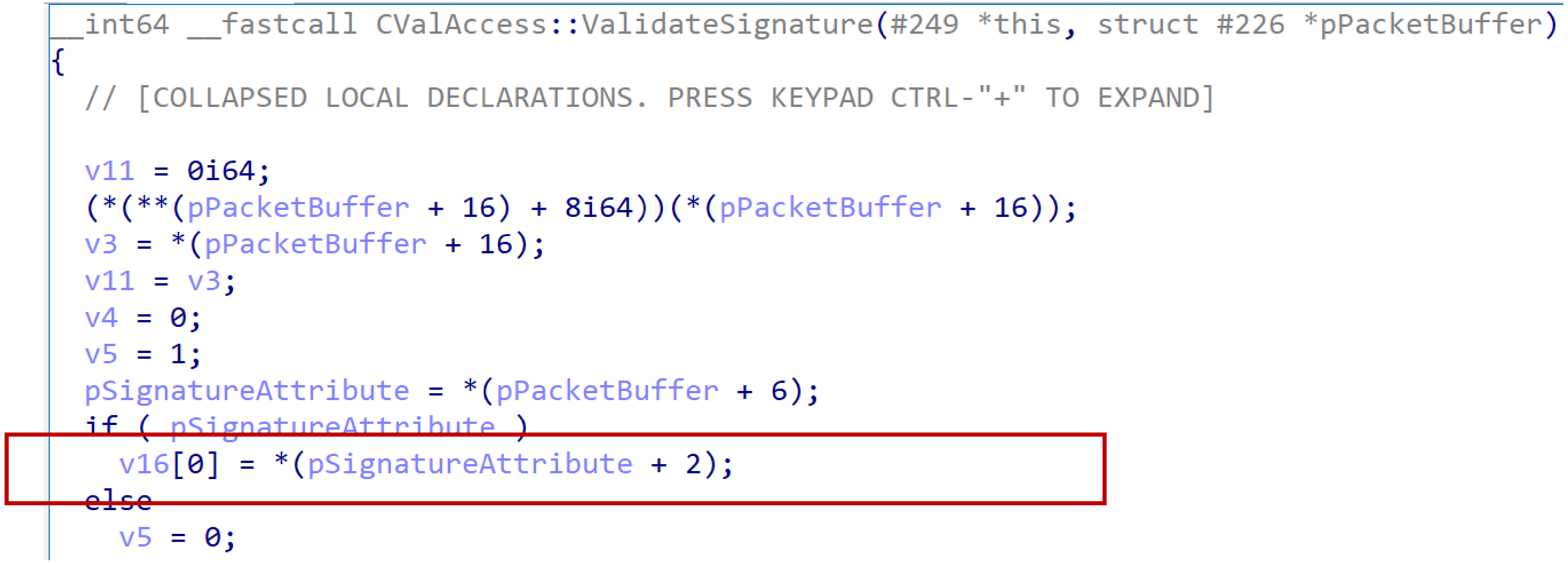
Authentication: 1812,1645

Authentication default ports 1812,1645

Accounting: 1813,1646

Accounting default ports 1813,1646

Case Study – NPS Packet Signature Attribute Pre-Auth Remote Info Leak



```
__int64 __fastcall CValAccess::ValidateSignature(#249 *this, struct #226 *pPacketBuffer)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v11 = 0i64;
  (*(**(pPacketBuffer + 16) + 8i64))(*(pPacketBuffer + 16));
  v3 = *(pPacketBuffer + 16);
  v11 = v3;
  v4 = 0;
  v5 = 1;
  pSignatureAttribute = *(pPacketBuffer + 6);
  if ( pSignatureAttribute )
    v16[0] = *(pSignatureAttribute + 2);
  else
    v5 = 0;
```

Read 16-bytes signature from the signature attribute in the packet, failed to validate the attribute size first

Case Study – NPS RADIUS Proxy Server auth DoS

```
void __fastcall RadiusProxyEngine::onReceive
{
 Attributes* pAttr = packet->pAttributes;
 DWORD dwAttrCount;
 …
 // We control pAttr->length, where is the DoS?
 for (; pAttr < pEnd; pAttr += pAttr->length)
     dwAttrCount  += pAttr->length;
 …
 }
```

Infinite loop if pAttr->length == 0

Tip: When playing with windows bounty, the pre-auth remote DoS/Info Leak attack scenarios are very high cost-performance, which you shouldn’t miss

Bug Collision – My Experience

- When our PPTP bug hunting was near to end, we started to have bug collisions with other researchers

  - Lucky we started early

- You don’t often find **high cost-performance target** in your bug hunting career

  - Never waste a chance
  - Never delay for even a single minute when you have a fruitful target
  - Personal Opinion: Less rest/holiday until you’ve dig enough, longer recovery/vacation after that for compensation

Result of PPTP Bug Hunting

- Found some pre-auth RCE bugs (race condition, classic memory corruption,…)
- Became familiar with the Windows RAS architecture and related kernel/user mode components
- And now we have enough confidence & motivation to look into other RAS VPN protocols

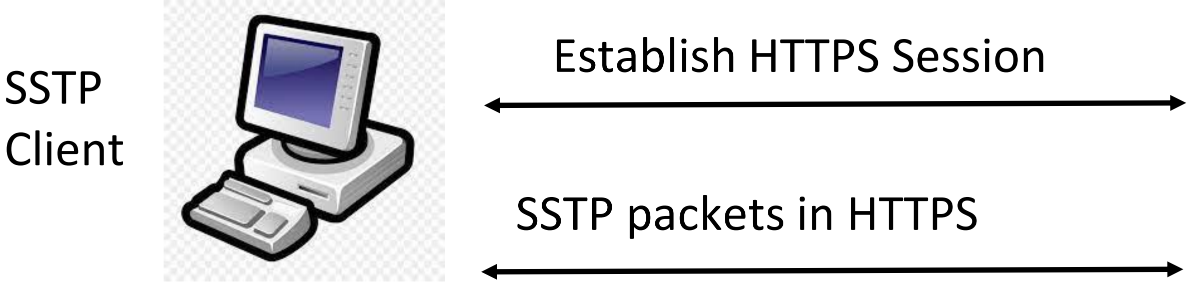
black hat USA 2023

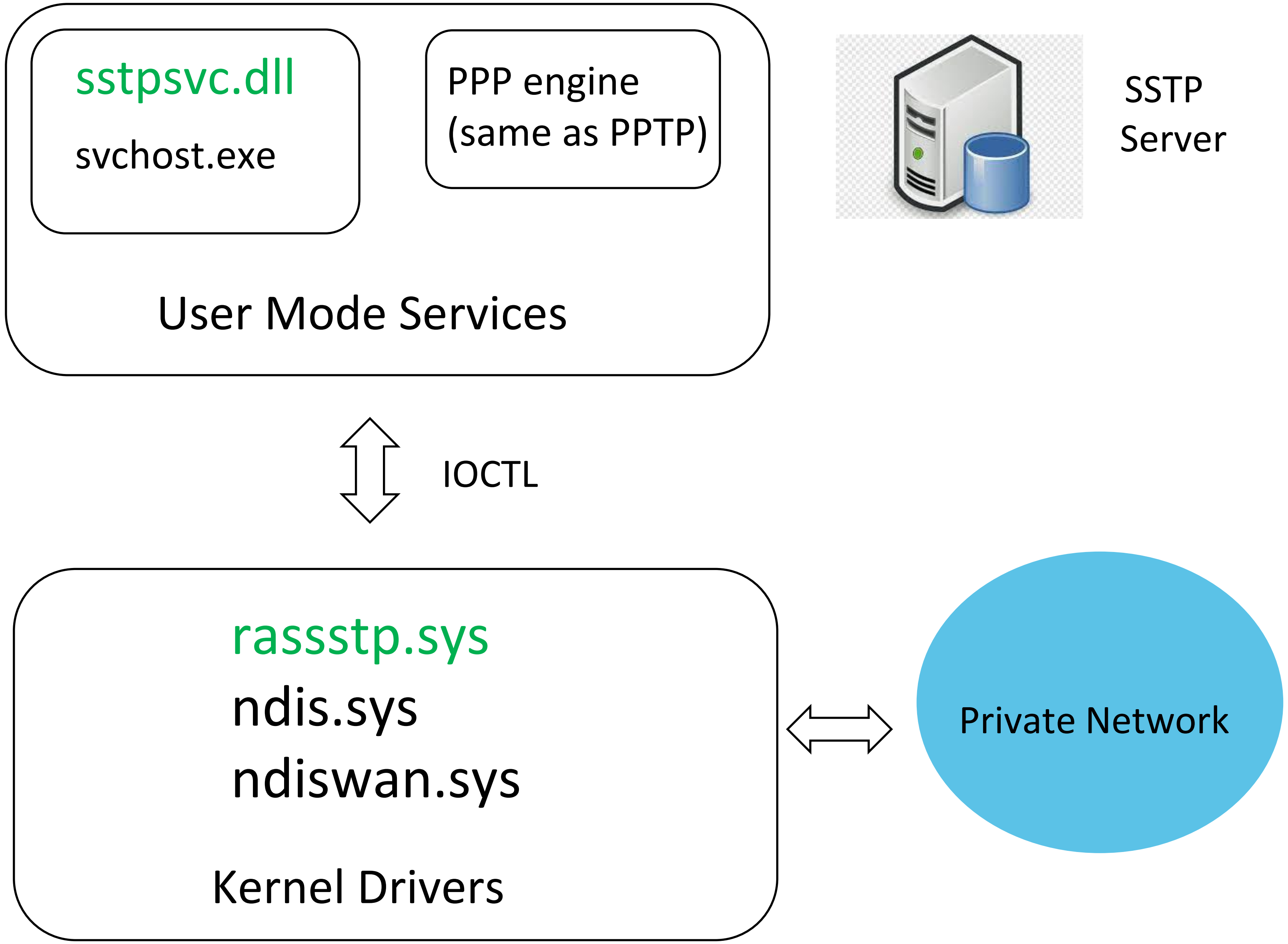
SSTP

SSTP - Secure Socket Tunneling Protocol

- Encapsulate Point-to-Point Protocol (PPP) traffic over HTTPS
- Developed by Microsoft
- Used in Azure point-to-site VPN

```
Header = ("""
SSTP_DUPLEX_POST /sra_{BA195980-CD49-458b-9E23-
C84EE0ADCD75}/ HTTP/1.1\r
Host: %s\r
SSTPCORRELATIONID: {62DFA5C0-E2E0-FD50-D286B00}\r
Content-Length: 4096\r
\r
""" % (HOST)).encode('utf-8')
```





0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5
Version								Reserved							C	LengthPacket									
Data (variable)																									
...																									

SSTP Packet

Pre-auth Attack Surfaces

- Kernel driver: rassstp.sys
- Handle SSTP packets, maintain life cycle of a SSTP call

- User mode service: sstpsvc.dll
- Maintain communication with client through HTTPs
- Communicate with kernel driver using IOCTL

- PPP engine: same as PPTP, won’t discuss again

rassstp.sys

- Handles SSTP messages
- Two types of packets: data packets and control packets

  Functionally similar to the PPTP data packets and control messages

- Maintains the whole life cycle of a SSTP session in kernel

  SstpContext - The most important data structure that represents a SSTP call session
- Same as PPTP, use ndis.sys/ndiswan.sys framework (whoops!)

rassstp.sys – Where to Look for Bugs?

- SSTP packet parsing, message handling

  - Control message

  - Data packet

- Session lifecycle management

  - Some similar designs to PPTP

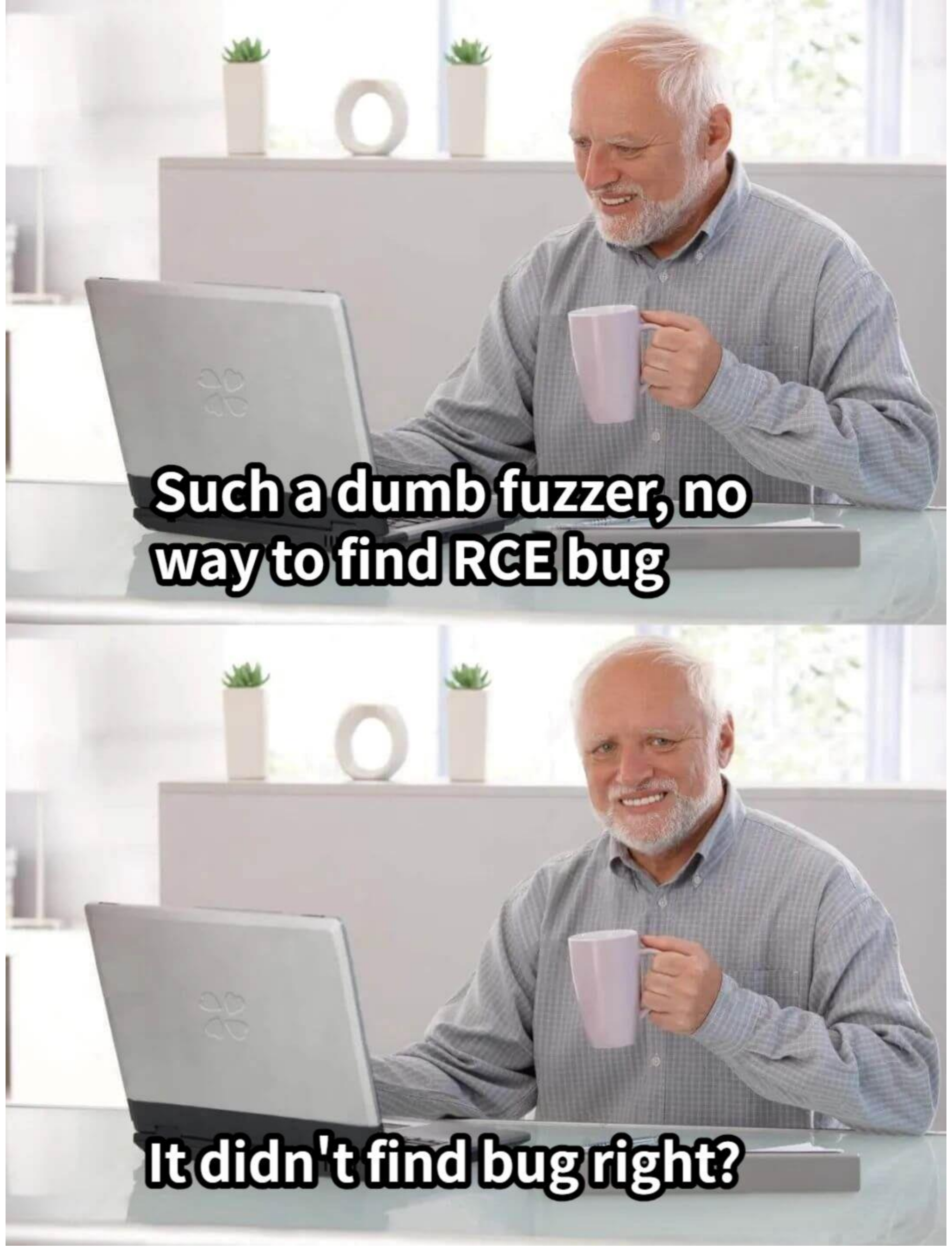
  - UAF/Race condition?

To Test Today's Luck

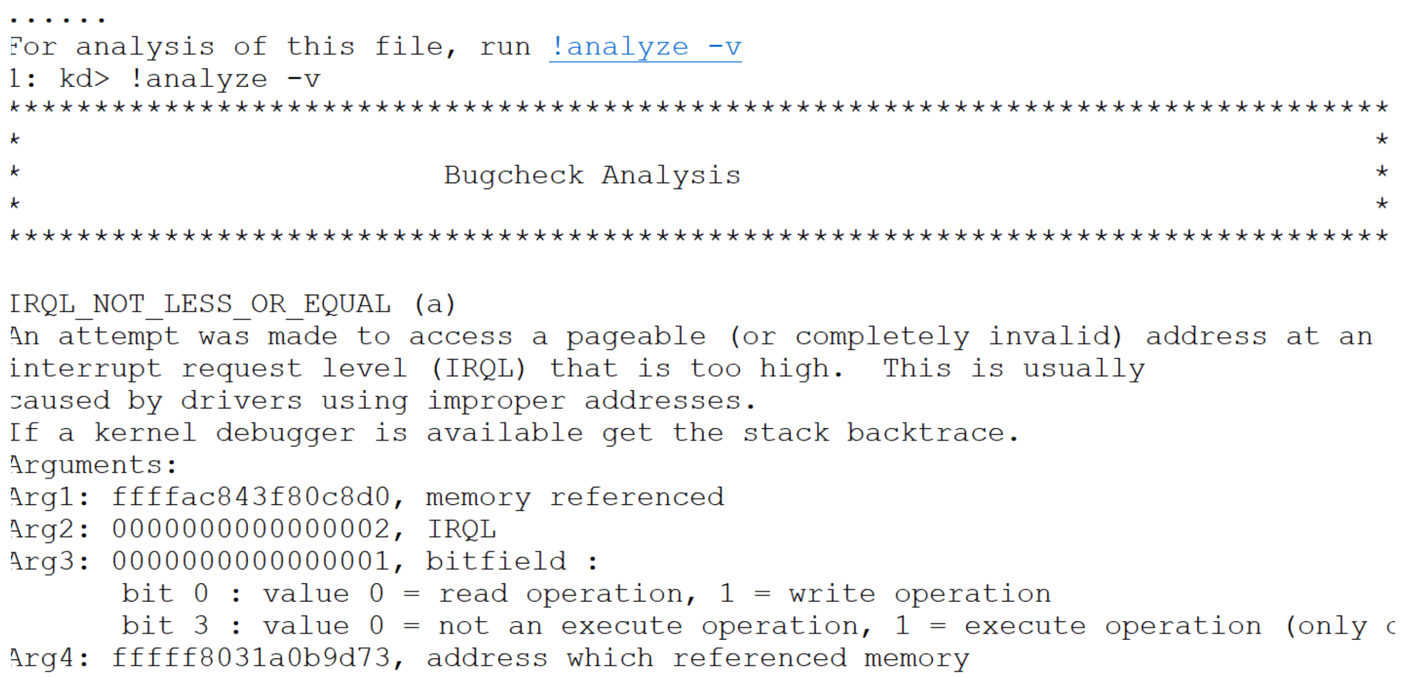
- Bought a lottery ticket
- Wrote a simple fuzzer in 20 minutes:

Create some threads, each thread just repeats:

1. Send a connect packet to the server
2. Sleep 0.5 seconds
3. Close the connection


Such a dumb fuzzer, no way to find RCE bug
It didn't find bug right?

The fuzzer triggered crash in 3 seconds, writing to freed memory



```
......
For analysis of this file, run !analyze -v
1: kd> !analyze -v
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

IRQL_NOT_LESS_OR_EQUAL (a)
An attempt was made to access a pageable (or completely invalid) address at an
interrupt request level (IRQL) that is too high.  This is usually
caused by drivers using improper addresses.
If a kernel debugger is available get the stack backtrace.
Arguments:
Arg1: ffffac843f80c8d0, memory referenced
Arg2: 0000000000000002, IRQL
Arg3: 0000000000000001, bitfield :
	bit 0 : value 0 = read operation, 1 = write operation
	bit 3 : value 0 = not an execute operation, 1 = execute operation (only c
Arg4: fffff8031a0b9d73, address which referenced memory
```

Didn’t win the lottery, btw

What’s Next

- Seems race condition also promising in SSTP

  Improved the fuzzer to support control/data messages, found several more bugs

- Meanwhile started manual auditing

  Found several race condition RCEs by manual auditing

Case Study – SstpContext Use After Free

- Found by auditing
- A common code pattern in rassstp.sys
- Try spotting the bug?

```
// Get SstpContext pointer from handle
Status = HfGetPointerFromHandle32(*(_QWORD *)(*(&WPP_MAIN_CB
+ 16) + 504), *pHandle, &pSstpContext);

// Increment ref counter to avoid SstpContext being freed
_InterlockedExchangeAdd(pSstpContext, 1u);
```

No lock protection, another thread can free SstpContext after HfGetPointerFromHandle and before _InterlockedExchangeAdd

Case Study – NdisVcHandle Use After Free

Thread 1 – Client sends a terminate request to PPP engine, the user mode service (RasMan)

tries to close the NdisVcHandle

```
ClIncomingCloseCall
{
// No Lock, no reference counter
Access NdisVcHandle
}
```

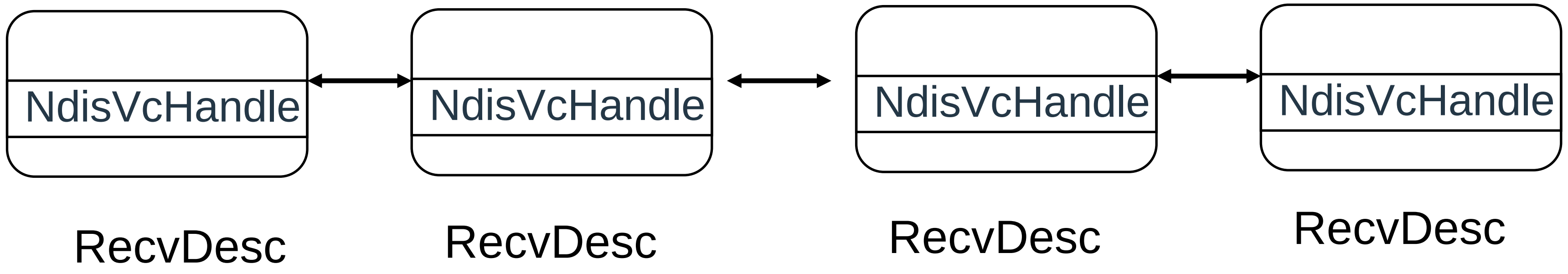
Race Window

Thread 2 – Close the socket, the kernel driver tries to clean up the SstpContext, which will delete the NdisVcHandle

```
InitiateSstpContextCleanup
{
// Free the NdisVcHandle
 NdisMCmDeleteVc(pCall->NdisVcHandle);
}
```

Case Study – Ndiswan.sys IoReceivePacket After Free

- Ndiswan.sys creates a RecvDesc structure for each data packet from client
- RecvDesc structure contains its’ corresponding NdisVcHandle (not ref counted!)
- User mode services calls into IoReceivePacket to get a packet data



```
IoReceivePacket()
{
.Find a RecvDesc
.Copy the packet data to user buffer
.Free the RecvDesc by NdisWanFreeRecvDesc
}
```

The NidsVcHandle can be freed in this race window, trigger UAF in NdisWanFreeRecvDesc when accessing RecvDesc->NdisHandle

Case Study – SSTP Timer UAF

- The first UAF found by our lucky fuzzer
- rassstp.sys maintains a global timer array

  Initialized when a new SSTP call comes in and if not yet initialized

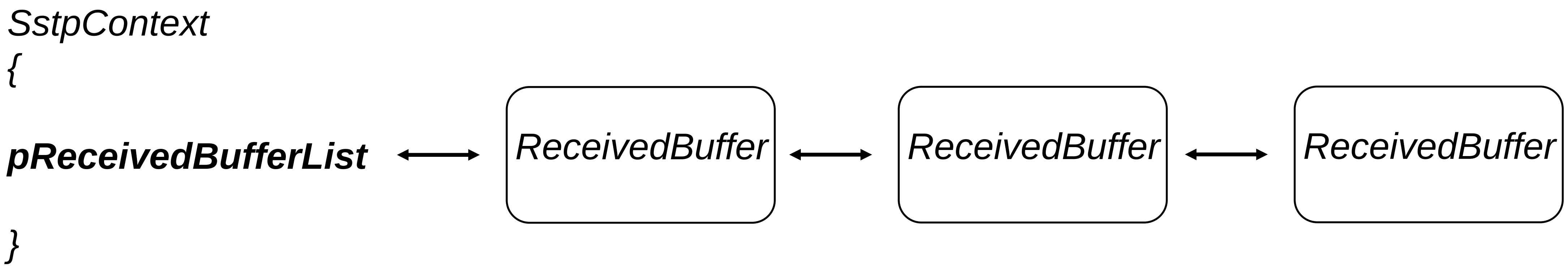
  Destroyed when there’s no existing call there
- The function SstpTimerInitialize accesses item in the timer array without lock protection, can access already destroyed timer item if race condition happens

User Mode Service (sstpsvc.dll)

- Relatively simple
- Dispatch packets between client and kernel driver most of the time
- Let’s still give it a try

Case Study – Sstpsvc Receive Buffer UAF

- For each packet from client, the service creates a ReceivedBuffer structure and links them in a list



SstpContext
{

pReceivedBufferList ⟷ *ReceivedBuffer* ⟷ *ReceivedBuffer* ⟷ *ReceivedBuffer*

}

Thread 1 – Worker thread

```
ProcessReceivedBytesWorker
{
EnterCriticalSection(SstpCtx->Lock)
ReceivedBuffer = RemoveList(SstpCtx->ReceivedBufferList)
LeaveCriticalSection(SstpCtx->Lock)

process ReceivedBuffer
}
```

Thread 2 – Close the session

```
InitiateCallContextCleanup
{
…
LeaveCriticalSection(SstpCtx->Lock)

For each ReceivedBuffer in SstpCtx-> ReceivedBufferList:
      Free ReceivedBuffer
}
```

No lock when freeing all ReceivedBuffer, triggering UAF in ProcessReceivedBytesWorker

Tips When looking for Kernel Race Bugs

- Focus on important resource

  - Draw a picture of the resource’s life cycle (creation/deletion/use)
  - Condition when accessed

    Spinlock/Reference counter/Callback/worker thread/Dispatch level?
  - Check if any of the 2 access places can race

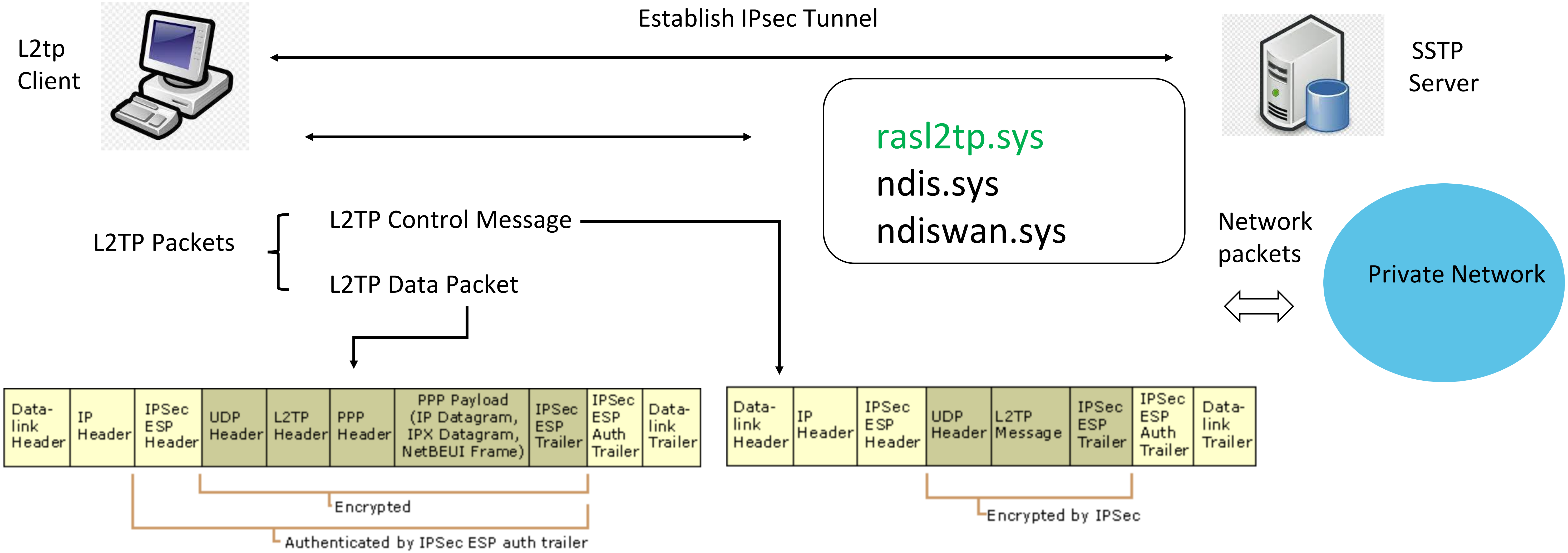
Result of SSTP Bug Hunting

- Found some pre-auth RCE bugs
- Race condition rocks again!
- Have more confidence & motivation to look into the rest RAS VPN protocols

L2TP

L2TP - Layer 2 Tunneling Protocol

- Usually rely on a IPsec tunnel to be established first
- After IPsec tunnel established, communicates with server through UDP port 1701, the UDP payload will be encrypted by IPsec



Establish IPsec Tunnel

L2tp Client

SSTP Server

rasl2tp.sys
ndis.sys
ndiswan.sys

L2TP Packets

L2TP Control Message

L2TP Data Packet

Network packets

Private Network

Data-link Header	IP Header	IPSec ESP Header	UDP Header	L2TP Header	PPP Header	PPP Payload (IP Datagram, IPX Datagram, NetBEUI Frame)	IPSec ESP Trailer	IPSec ESP Auth Trailer	Data-link Trailer

Encrypted

Authenticated by IPSec ESP auth trailer

Data-link Header	IP Header	IPSec ESP Header	UDP Header	L2TP Message	IPSec ESP Trailer	IPSec ESP Auth Trailer	Data-link Trailer

Encrypted by IPSec

Pre-auth Attack Surfaces

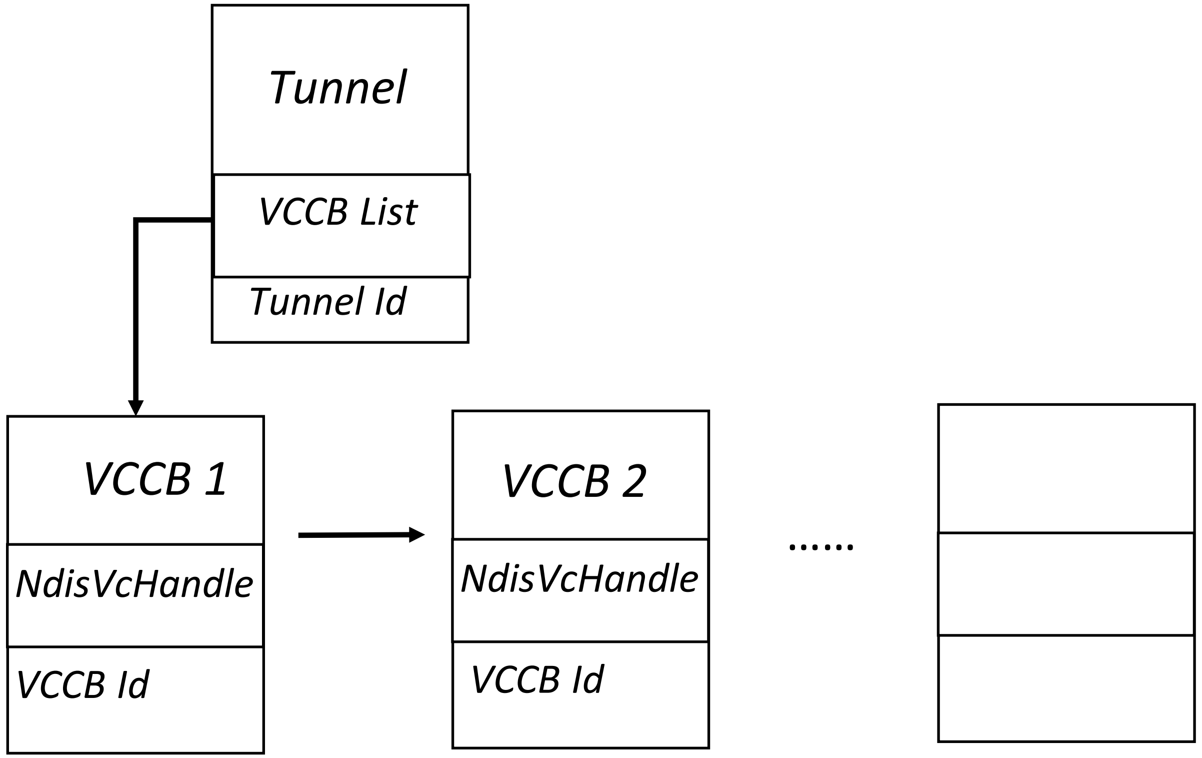
- In L2TP, there are two authentications:
- When establishing IPsec - Partial authentication

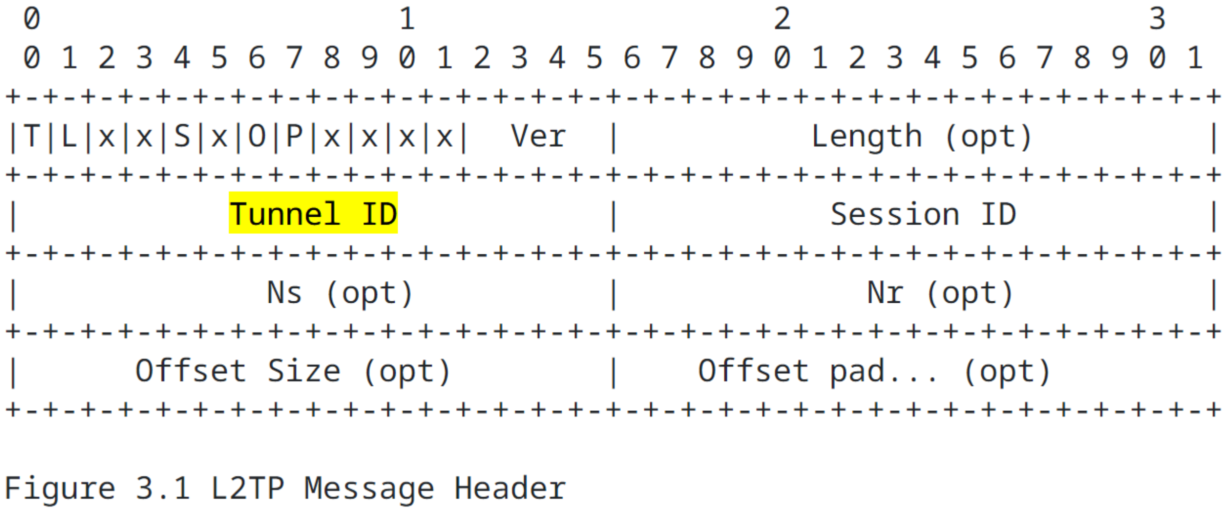
  Pre-shared Key(shared to multiple people)/Certificate
- When connect the VPN – Full authentication

- IPsec: ikeext.dll (will be discussed in IKEv1/IKE2 part)
- Before Full Authentication: rasl2tp.sys
- message parsing, tunnel/call life cycle management

Rasl2tp.sys - Key Data Structures

- *Tunnel*: Represents a l2tp control channel between client and server, similar to a *Control* in PPTP
- *VCCB:* Represents a l2tp call, similar to a *Call* in PPTP
- l2tp message header contains tunnel id/VCCB id to locate tunnel and VCCB

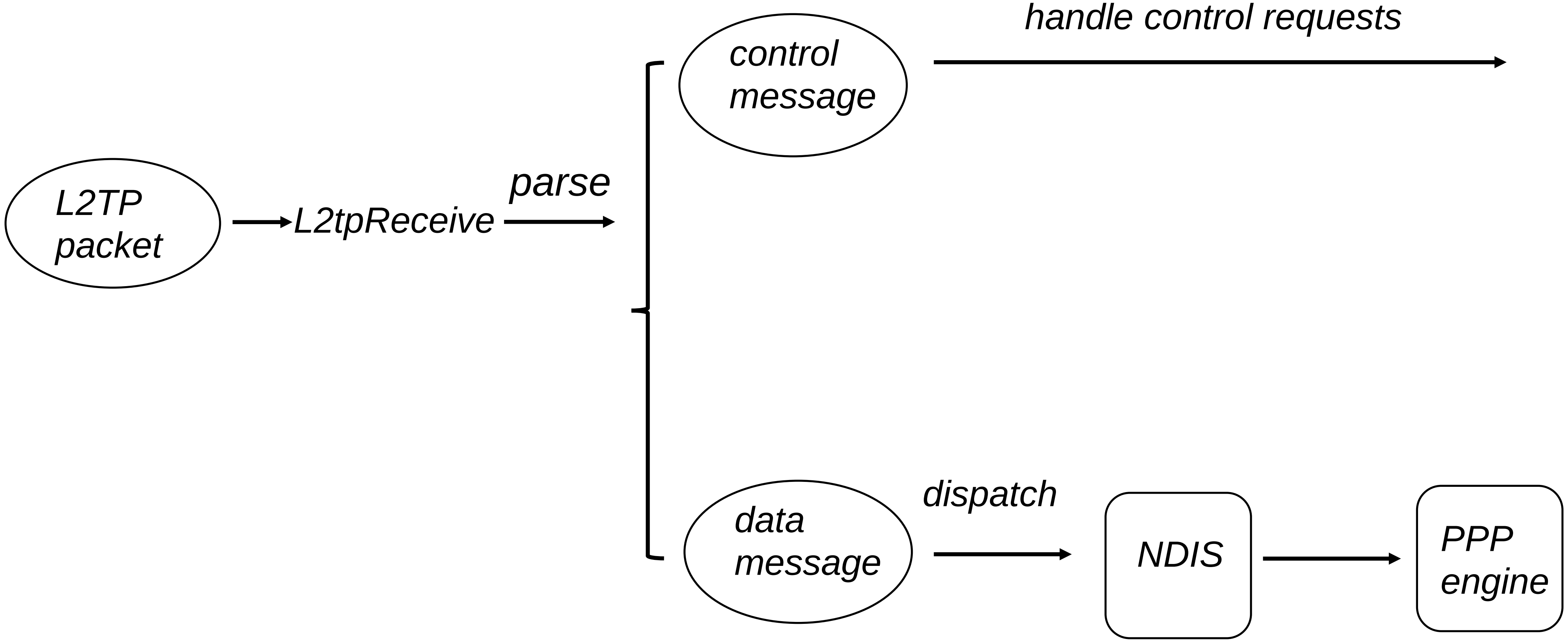




```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|T|L|x|x|S|x|O|P|x|x|x|x|  Ver  |          Length (opt)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Tunnel ID           |           Session ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Ns (opt)          |             Nr (opt)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Offset Size (opt)        |    Offset pad... (opt)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 3.1 L2TP Message Header
```

L2tpReceive – L2TP Packet Entry Point

- 2 types of L2TP message: control message and data message



L2TP packet

L2tpReceive

parse

control message

handle control requests

data message

dispatch

NDIS

PPP engine

```
   0  (reserved)

   1  (SCCRQ)    Start-Control-Connection-Request
   2  (SCCRP)    Start-Control-Connection-Reply
   3  (SCCCN)    Start-Control-Connection-Connected
   4  (StopCCN)  Stop-Control-Connection-Notification
   5  (reserved)
   6  (HELLO)    Hello

Call Management

   7  (OCRQ)     Outgoing-Call-Request
   8  (OCRP)     Outgoing-Call-Reply
   9  (OCCN)     Outgoing-Call-Connected
   10 (ICRQ)     Incoming-Call-Request
   11 (ICRP)     Incoming-Call-Reply
   12 (ICCN)     Incoming-Call-Connected
   13 (reserved)
   14 (CDN)      Call-Disconnect-Notify

Error Reporting

   15 (WEN)      WAN-Error-Notify
```

Rasl2tp.sys – Where to look for bugs?

- Message parsing/processing

  - State machine

  - Control message

  - Data message

- Key objects Life cycle

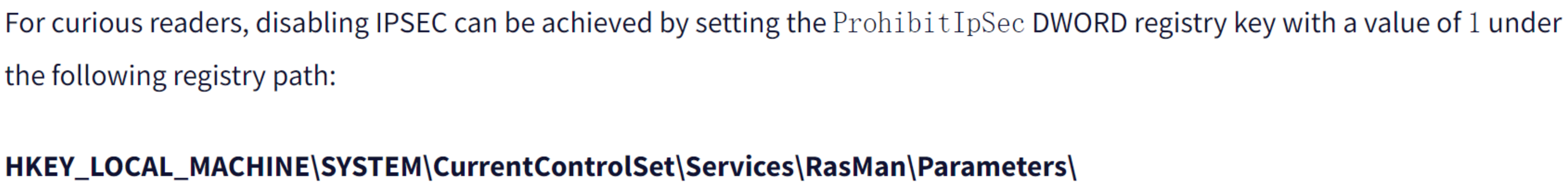
  - Tunnel & VCCB

  - UAF/Race Condition?

A Quick Fuzzer as Usual

- Multiple threads
- L2TP message mutating based on coverage
- Random control message sequence

- A very useful tip shared by Alex Nicols (@i4mchr00t) when fuzzing rasl2tp.sys

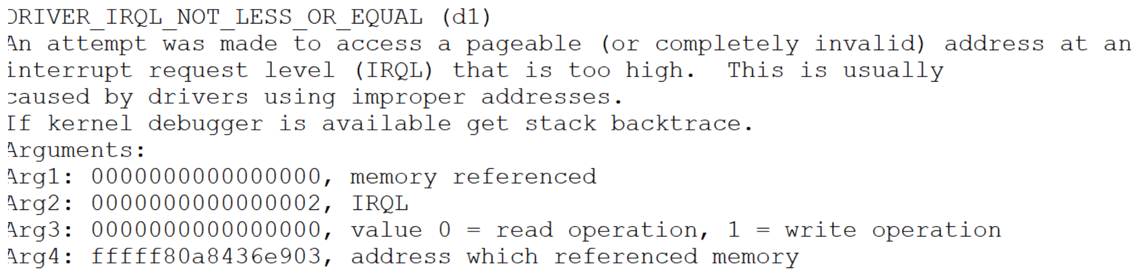


For curious readers, disabling IPSEC can be achieved by setting the `ProhibitIpSec` DWORD registry key with a value of `1` under the following registry path:

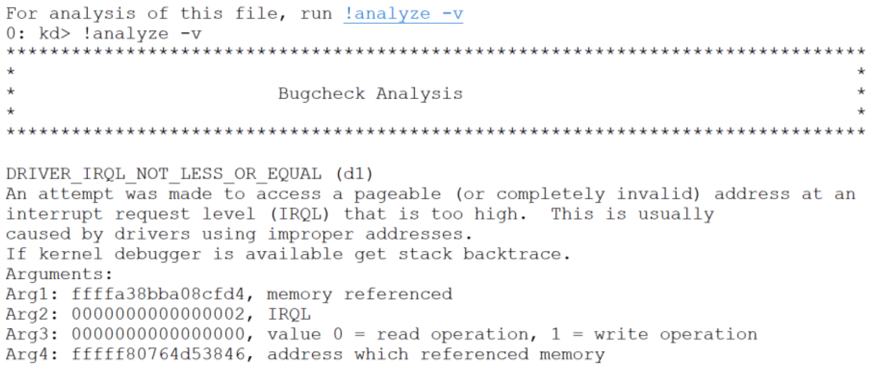
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\Parameters\

Fuzz Result

- Got a few crashes (UAF, NPD)
- Switched to manual auditing and found several others



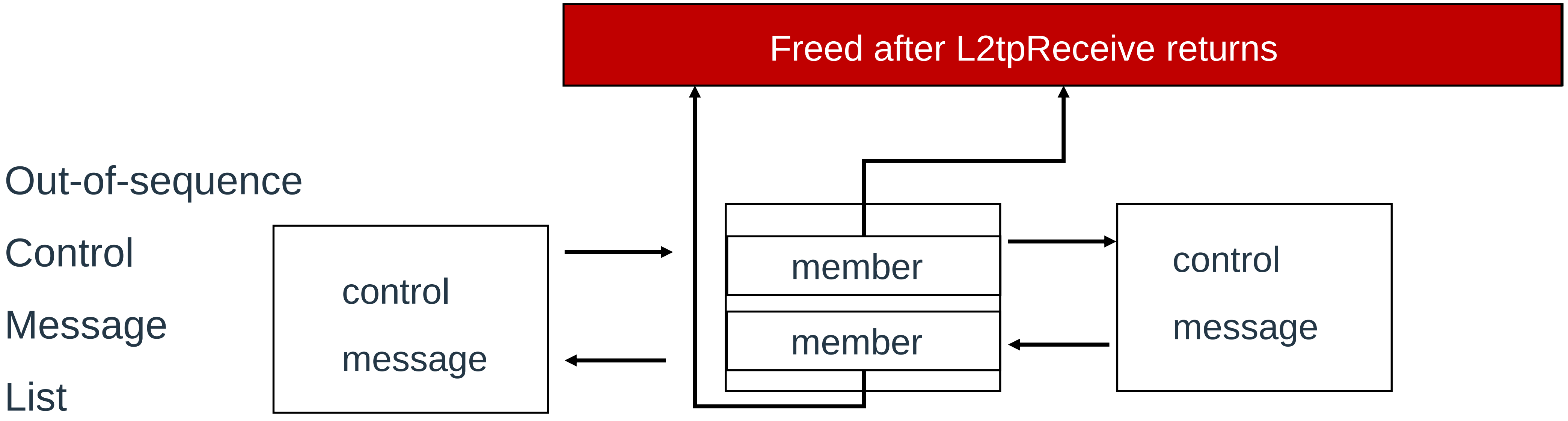
```
DRIVER_IRQL_NOT_LESS_OR_EQUAL (d1)
An attempt was made to access a pageable (or completely invalid) address at an
interrupt request level (IRQL) that is too high.  This is usually
caused by drivers using improper addresses.
If kernel debugger is available get stack backtrace.
Arguments:
Arg1: 0000000000000000, memory referenced
Arg2: 0000000000000002, IRQL
Arg3: 0000000000000000, value 0 = read operation, 1 = write operation
Arg4: fffff80a8436e903, address which referenced memory
```



```
For analysis of this file, run !analyze -v
0: kd> !analyze -v
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

DRIVER_IRQL_NOT_LESS_OR_EQUAL (d1)
An attempt was made to access a pageable (or completely invalid) address at an
interrupt request level (IRQL) that is too high.  This is usually
caused by drivers using improper addresses.
If kernel debugger is available get stack backtrace.
Arguments:
Arg1: ffffa38bba08cfd4, memory referenced
Arg2: 0000000000000002, IRQL
Arg3: 0000000000000000, value 0 = read operation, 1 = write operation
Arg4: fffff80764d53846, address which referenced memory
```

Case Study – Control Message UAF

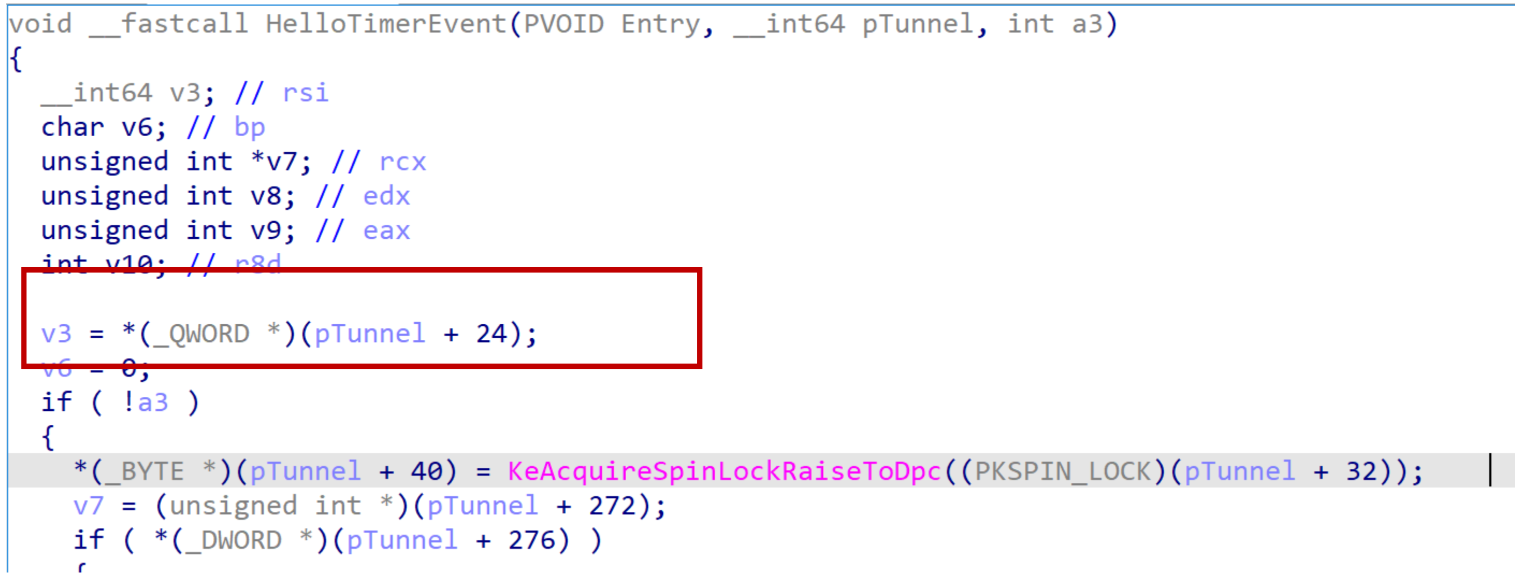
- Found by fuzzer
- Out-of-sequence control messages will be copied and insert into a list, incorrectly using of shallow copy leads to UAF



Case Study – Tunnel UAF in Timer Event

- Found by fuzzer
- Race condition between a timer callback and tunnel closing

TimerQScheduleItem(HelloTimerEvent) ⇒



```
void __fastcall HelloTimerEvent(PVOID Entry, __int64 pTunnel, int a3)
{
  __int64 v3; // rsi
  char v6; // bp
  unsigned int *v7; // rcx
  unsigned int v8; // edx
  unsigned int v9; // eax
  int v10; // r8d

  v3 = *(_QWORD *)(pTunnel + 24);
  v6 = 0;
  if ( !a3 )
  {
    *(_BYTE *)(pTunnel + 40) = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(pTunnel + 32));
    v7 = (unsigned int *)(pTunnel + 272);
    if ( *(_DWORD *)(pTunnel + 276) )
    {
```

Accessing tunnel without lock,

another thread can free the tunnel

Case Study – VCCB Use After Free

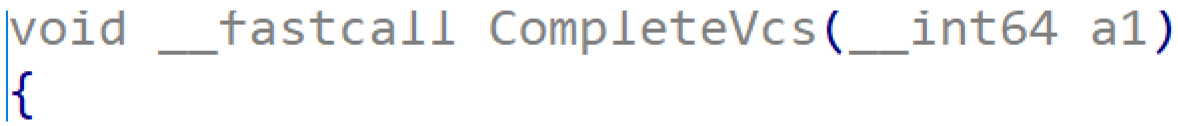
Client send a Incoming-Call-Request (ICRQ)

```
SetupVcAsynchronously
{
LcmCmCreateVc(…,&pVc)
KeAcquireSpinLockRaiseToDpc
InsertTailList(pTunnel->vcList, &pVc->listEntry)
KeReleaseSpinLock
// No Lock protection
Access pVc
}
```

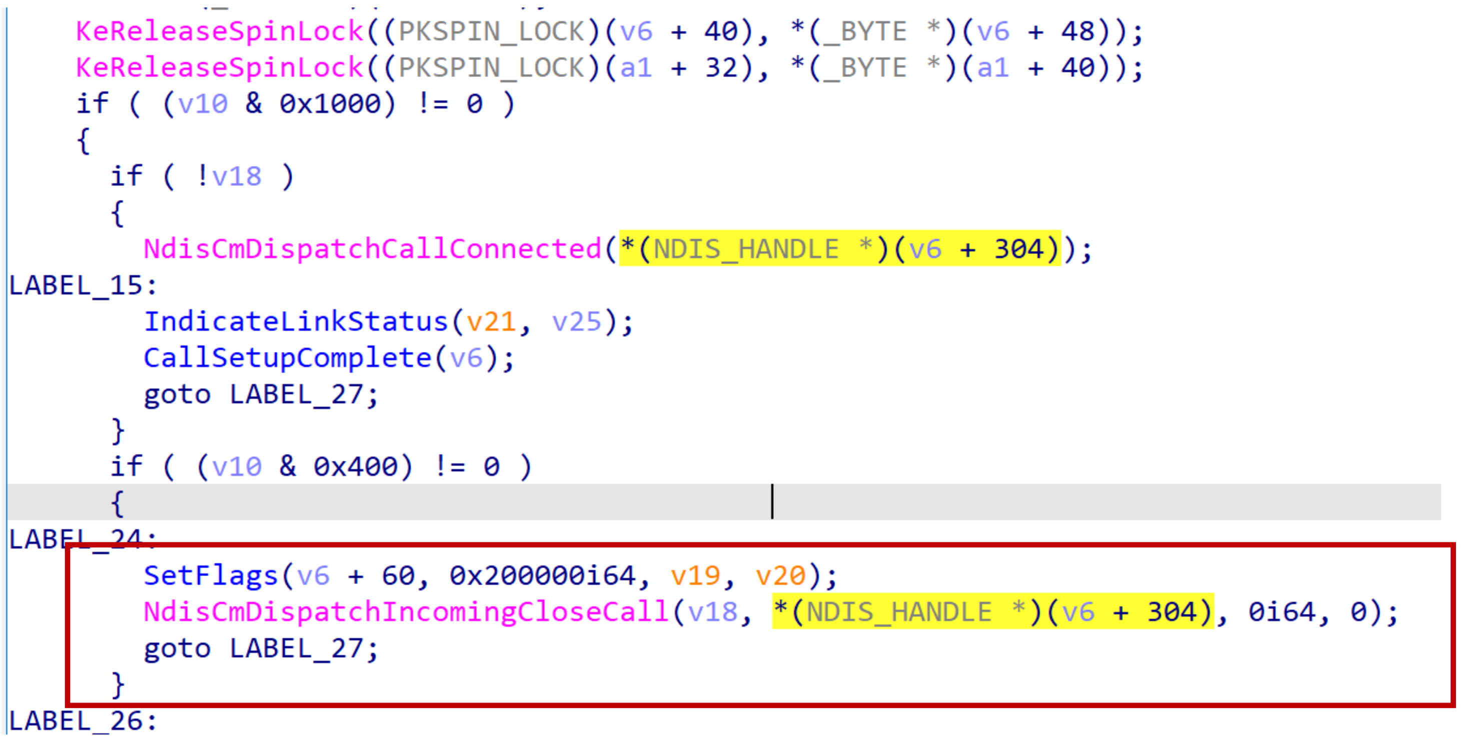
Another thread closes the tunnel which resulting in freeing all VCs in this race window

Case Study – NdisVcHandle Use After Free

- NdisVcHandle – Our old friend who already brought us many bugs in PPTP/SSTP
- Proves again the difficulty of handling across references across multiple modules



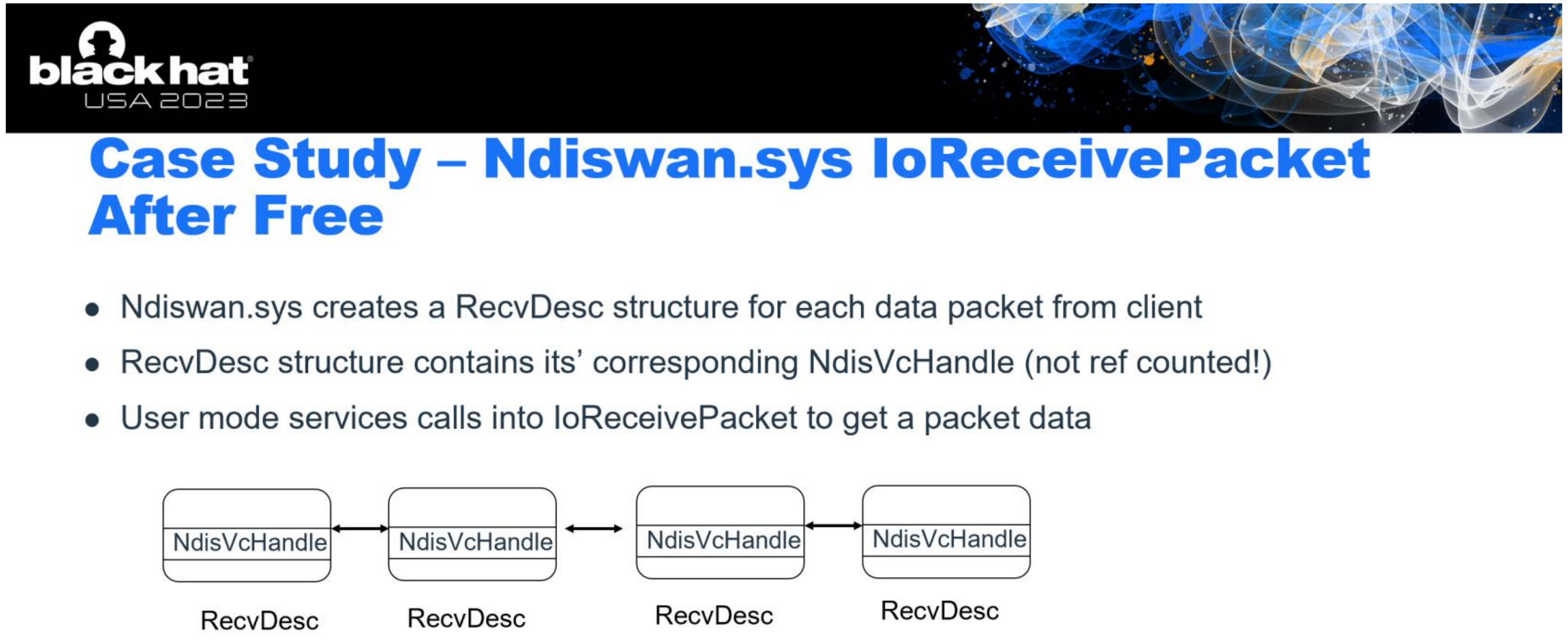
```
void __fastcall CompleteVcs(__int64 a1)
{
```



```
    KeReleaseSpinLock((PKSPIN_LOCK)(v6 + 40), *(_BYTE *)(v6 + 48));
    KeReleaseSpinLock((PKSPIN_LOCK)(a1 + 32), *(_BYTE *)(a1 + 40));
    if ( (v10 & 0x1000) != 0 )
    {
      if ( !v18 )
      {
        NdisCmDispatchCallConnected(*(NDIS_HANDLE *)(v6 + 304));
LABEL_15:
        IndicateLinkStatus(v21, v25);
        CallSetupComplete(v6);
        goto LABEL_27;
      }
      if ( (v10 & 0x400) != 0 )
      {
LABEL_24:
        SetFlags(v6 + 60, 0x200000i64, v19, v20);
        NdisCmDispatchIncomingCloseCall(v18, *(NDIS_HANDLE *)(v6 + 304), 0i64, 0);
        goto LABEL_27;
      }
LABEL_26:
```

A disconnection from user mode RasMan service can free the NdisVcHandle

Case Study – NdisWan RecvDesc Use After Free



black hat USA 2023

Case Study – Ndiswan.sys IoReceivePacket After Free

- Ndiswan.sys creates a RecvDesc structure for each data packet from client
- RecvDesc structure contains its' corresponding NdisVcHandle (not ref counted!)
- User mode services calls into IoReceivePacket to get a packet data

NdisVcHandle ↔ NdisVcHandle ↔ NdisVcHandle ↔ NdisVcHandle

RecvDesc RecvDesc RecvDesc RecvDesc

Same bug, different component

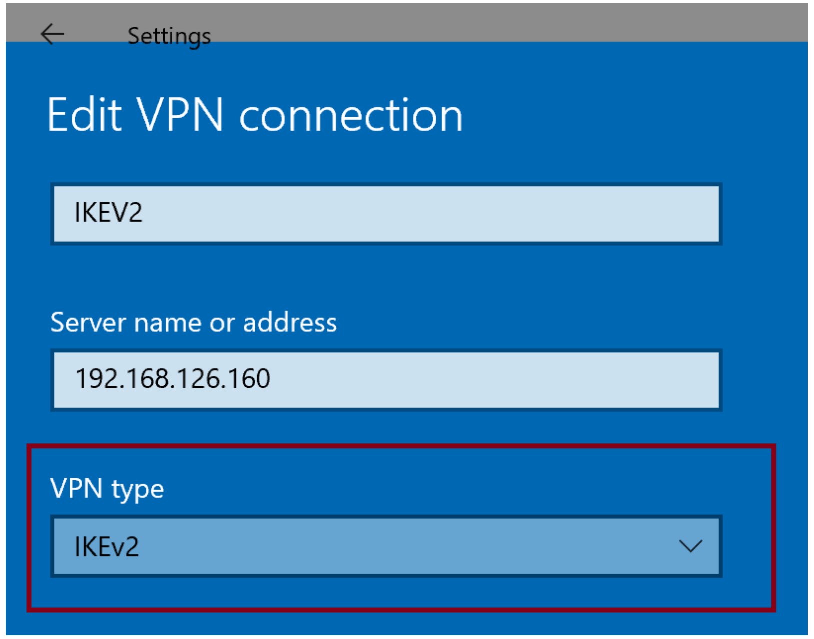
Result of L2TP Bug Hunting

- Found some bugs in L2TP kernel driver
- All bugs are triggered after the IPsec tunnel established

IKE

IKE - Internet Key Exchange

- IKE: an authentication protocol to establish security tunnel for IPsec
- Two versions: IKEv1 and IKEv2, IKEv2 has many improvements over IKEv1
- Windows L2TP VPN can use IKEv1 for tunnel establishing
- Built-in IKEv2 VPN in Windows
  - – IKEv2 + agilevpn.sys kernel driver
  - – Used in Azure Site-to-Site VPN



Edit VPN connection

IKEV2

Server name or address

192.168.126.160

VPN type

IKEv2

IKE – Pre-auth Attack Surfaces

- Everything before authentication finish

  - IKE packets parsing/processing

  - UAF/Race condition
- IKEv1: UDP port 4500
- IKEv2: UDP port 500
- ikeext.dll

Learn the SPEC

Exhausted.

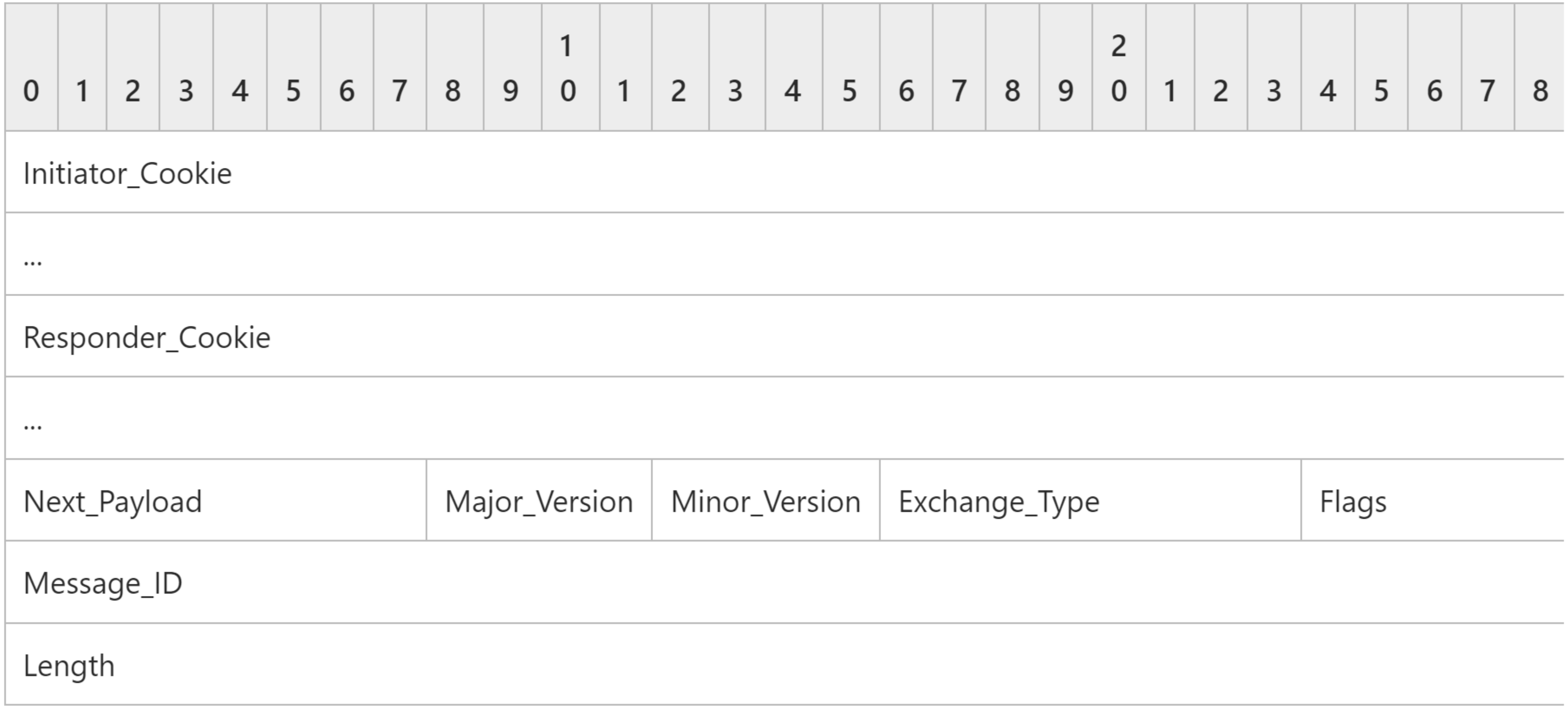
Just fuzz it.

A Mutation Based Fuzzer

- Mutate normal IKEv1/IKEv2 messages
- Shuffle the message sequence
- Multi-threaded to check UAF/Race Condition
- Gives a few crashes (NPD, OOB) – at least now we have something to crash to server
- Use the crash PoC as entries, manually audit and understand the protocol slowly

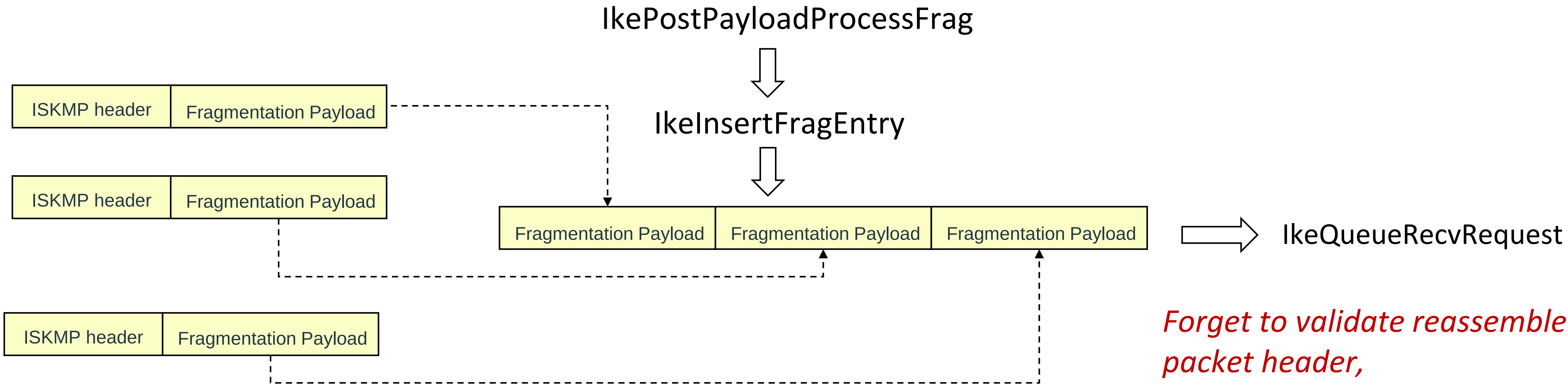
Case Study – IKEv1 Fragmentation RCE

- IKE supports packet fragmentation – our favorite feature in a network protocols



0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8
Initiator_Cookie																												
…																												
Responder_Cookie																												
…																												
Next_Payload								Major_Version				Minor_Version				Exchange_Type								Flags				
Message_ID																												
Length																												

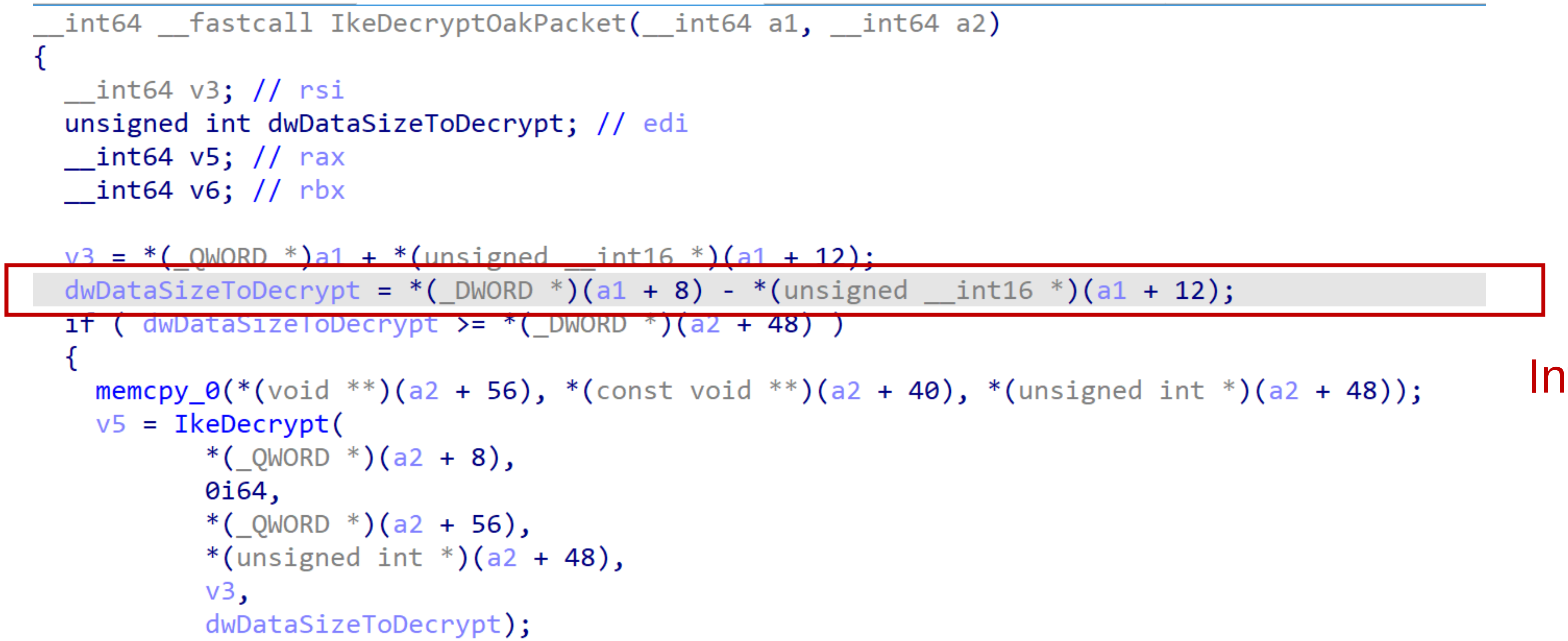
ISAKMP header



Forget to validate reassembled packet header,
resulting in OOB Read/Write

Case Study – IKEv1 IkeDecryptOakPacket Integer Overflow

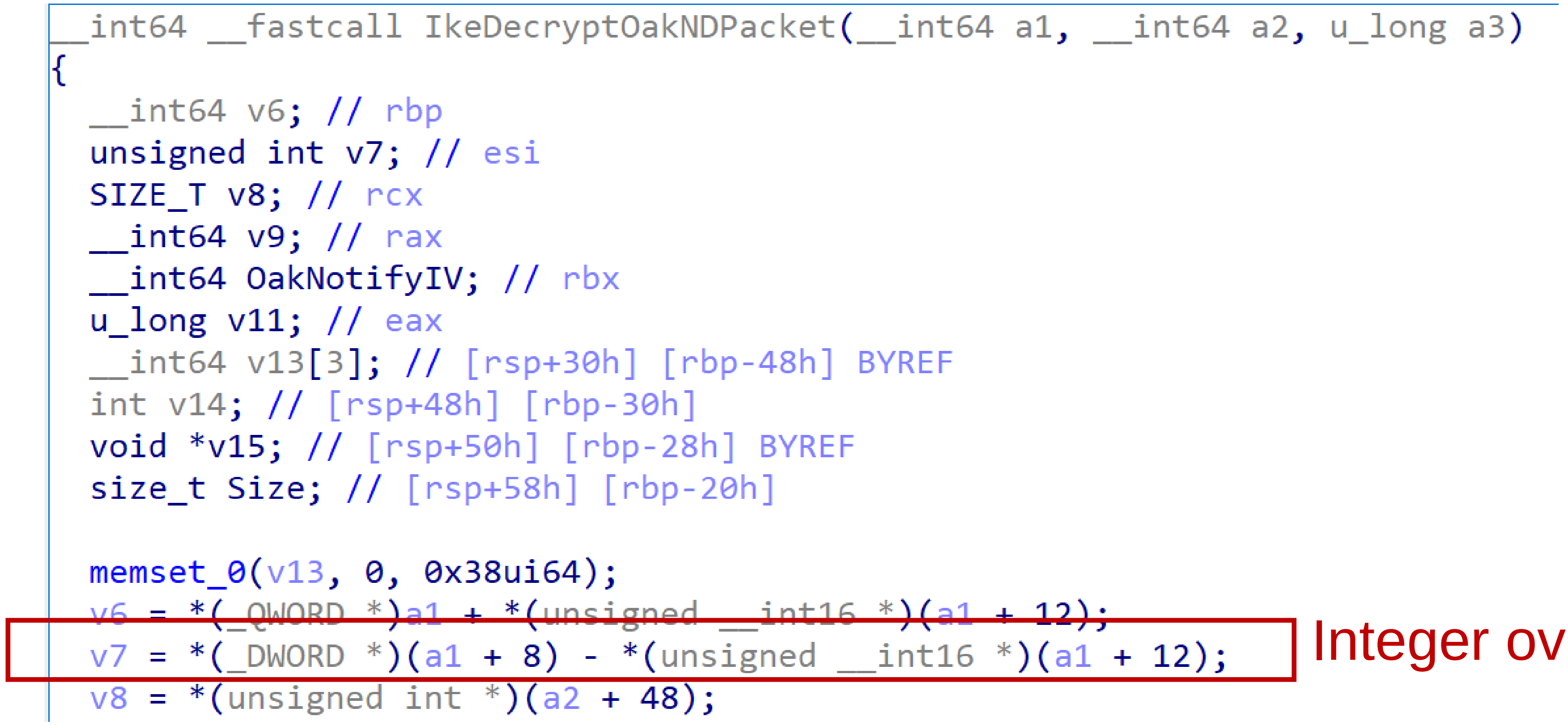
- Classic integer overflow resulting in OOB write, when decrypting encrypted IKE packet



```
__int64 __fastcall IkeDecryptOakPacket(__int64 a1, __int64 a2)
{
  __int64 v3; // rsi
  unsigned int dwDataSizeToDecrypt; // edi
  __int64 v5; // rax
  __int64 v6; // rbx

  v3 = *(_QWORD *)a1 + *(unsigned __int16 *)(a1 + 12);
  dwDataSizeToDecrypt = *(_DWORD *)(a1 + 8) - *(unsigned __int16 *)(a1 + 12);
  if ( dwDataSizeToDecrypt >= *(_DWORD *)(a2 + 48) )
  {
    memcpy_0(*(void **)(a2 + 56), *(const void **)(a2 + 40), *(unsigned int *)(a2 + 48));
    v5 = IkeDecrypt(
           *(_QWORD *)(a2 + 8),
           0i64,
           *(_QWORD *)(a2 + 56),
           *(unsigned int *)(a2 + 48),
           v3,
           dwDataSizeToDecrypt);
```

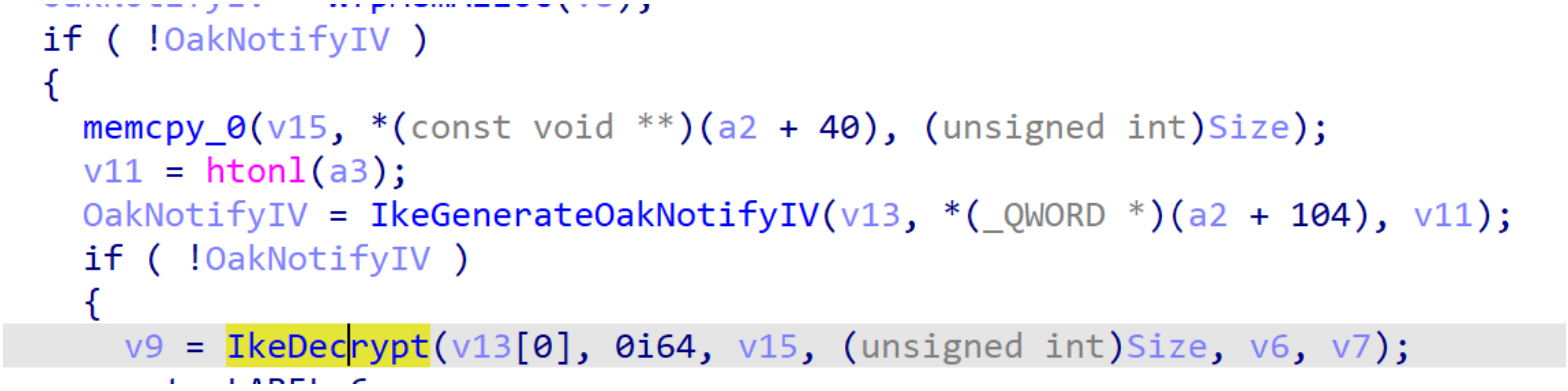
Integer overflow

Another similar Bug in Another Function



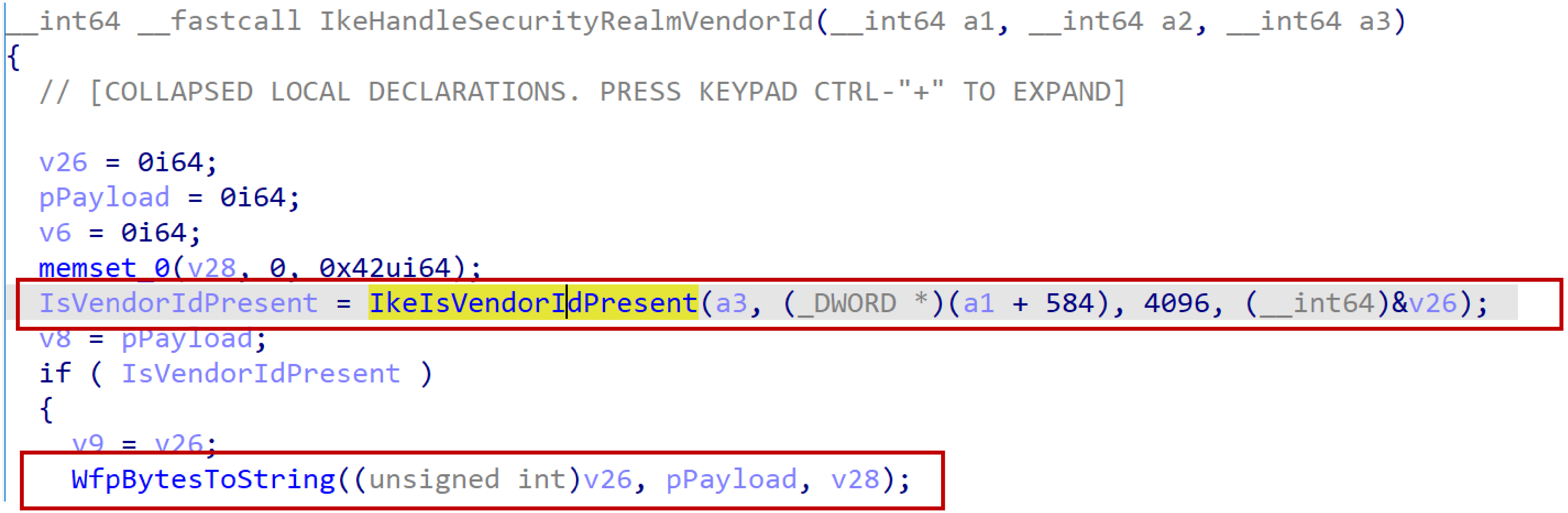
```
__int64 __fastcall IkeDecryptOakNDPacket(__int64 a1, __int64 a2, u_long a3)
{
  __int64 v6; // rbp
  unsigned int v7; // esi
  SIZE_T v8; // rcx
  __int64 v9; // rax
  __int64 OakNotifyIV; // rbx
  u_long v11; // eax
  __int64 v13[3]; // [rsp+30h] [rbp-48h] BYREF
  int v14; // [rsp+48h] [rbp-30h]
  void *v15; // [rsp+50h] [rbp-28h] BYREF
  size_t Size; // [rsp+58h] [rbp-20h]

  memset_0(v13, 0, 0x38ui64);
  v6 = *(_QWORD *)a1 + *(unsigned __int16 *)(a1 + 12);
  v7 = *(_DWORD *)(a1 + 8) - *(unsigned __int16 *)(a1 + 12);
  v8 = *(unsigned int *)(a2 + 48);
```

Integer overflow



```
 if ( !OakNotifyIV )
 {
   memcpy_0(v15, *(const void **)(a2 + 40), (unsigned int)Size);
   v11 = htonl(a3);
   OakNotifyIV = IkeGenerateOakNotifyIV(v13, *(_QWORD *)(a2 + 104), v11);
   if ( !OakNotifyIV )
   {
     v9 = IkeDecrypt(v13[0], 0i64, v15, (unsigned int)Size, v6, v7);
```

Case Study – IKEv2 NPD Pre-auth DoS



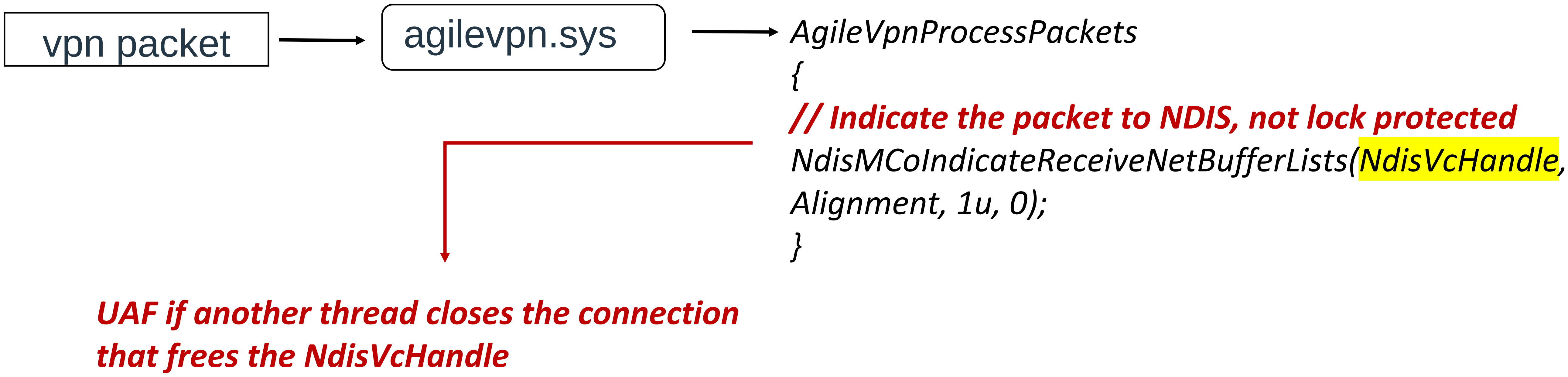
```
__int64 __fastcall IkeHandleSecurityRealmVendorId(__int64 a1, __int64 a2, __int64 a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v26 = 0i64;
  pPayload = 0i64;
  v6 = 0i64;
  memset_0(v28, 0, 0x42ui64);
  IsVendorIdPresent = IkeIsVendorIdPresent(a3, (_DWORD *)(a1 + 584), 4096, (__int64)&v26);
  v8 = pPayload;
  if ( IsVendorIdPresent )
  {
    v9 = v26;
    WfpBytesToString((unsigned int)v26, pPayload, v28);
```

IkeIsVendorIdPresent only verifies that there is a VendorId payload exists, However the payload data can be empty (zero-size)

Null-Pointer-Deference when VendorId payload's data size is 0

Case Study – IKEv2 AgileVPN NdisVcHandle Race Condition UAF

- Achievement unlocked: Ndis handle UAF in every windows VPN protocol we researched 🤦



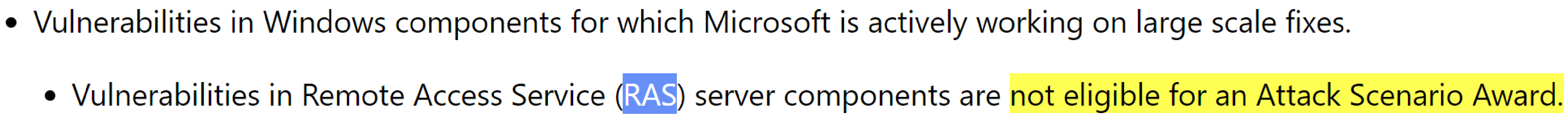
```
AgileVpnProcessPackets
{
// Indicate the packet to NDIS, not lock protected
NdisMCoIndicateReceiveNetBufferLists(NdisVcHandle,
Alignment, 1u, 0);
}
```

UAF if another thread closes the connection that frees the NdisVcHandle

Result of IKE Bug Hunting

- We found pre-auth RCE/DoS bugs in IKE
- Not as many bugs as other protocols
- Based on the result, IKEv2 is more secure than others, choose IKEv2 if you have to use a Windows VPN (standalone server/Azure cloud)

A Medal from Microsoft for Your Bad-ass Work



- Vulnerabilities in Windows components for which Microsoft is actively working on large scale fixes.
  - Vulnerabilities in Remote Access Service (RAS) server components are not eligible for an Attack Scenario Award.

Q: Similarities between a productive bug hunter and a playboy?

They move to next target fast

black hat®
USA 2023

Future Work & Take Aways

Future Work

- Explore other Windows remote pre-auth attack surfaces

  - Authentication protocols/methods

  - Network protocols (IPsec, TCP/IP, Peer-to-peer, Bluetooth, Wireless…)

  - Other components in RAS (Routing, …)

  - Services in domain environment (LDAP, …)

Take Aways

- The implementation of Windows RAS VPN protocols are complex, with both kernel drivers and user mode services, making them good targets for looking for remote pre-auth bugs
- Don’t forget to try race condition when researching windows remote protocols
- Use both fuzzing and manual auditing, avoid only relying on one method
- Be smart, eager and greedy in bug hunting
- We wish every one a big success in your bug hunting journey

Thanks!

CYBER KUNLUN

@guhe120

https://www.cyberkl.com/en