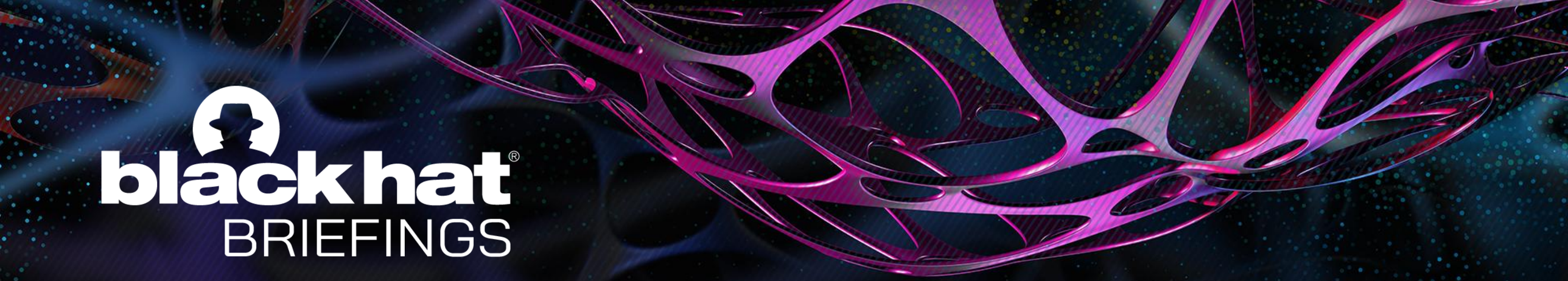


AUGUST 6-7, 2025

MANDALAY BAY / LAS VEGAS

Diving into Windows HTTP: Unveiling Hidden Preauth Vulnerabilities in Windows HTTP Services

Qibo Shi(k0shl), VictorV, Wei Xiao, Zhiniang Peng

About us

Qibo Shi(k0shl) | Senior Security Researcher of Cyber Kunlun Lab

VictorV | Senior Security Researcher of Cyber Kunlun Lab

Wei Xiao | Senior Security Researcher of Cyber Kunlun Lab

Zhiniang Peng | Associate Professor of Huazhong University of Science and Technology

Agenda

I. Background

II. Overview of the Windows HTTP Service Framework

III. Exploring Logic Flaws Leading to Pre-auth DoS

IV. Parsing and Handling Stages Leading to Pre-auth RCE

V. Conclusion

black hat BRIEFINGS

AUGUST 6-7, 2025

MANDALAY BAY / LAS VEGAS

Background

Why HTTP Services?

- ✓ Most of them are unauthenticated.
- ✓ No user interaction required.
- ✓ No additional configuration needed.
- ✓ Few researchers have focused on it before.
- ✓ Many Windows Services rely on the Windows HTTP APIs (httpapi.dll).

Overview of HTTP Services in Windows

- HttpCreateServerSession https://learn.microsoft.com/en-us/windows/win32/api/http/nf-http-httpcreateserversession

  Initializes a new HTTP Server API session.

  This is the starting point for configuring a server-side HTTP stack.

- HttpAddUrl/HttpAddUrlToUrlGroup https://learn.microsoft.com/en-us/windows/win32/api/http/nf-http-httpaddurl

  Registers a URL to listen on.

  Binds a specific URL to the server session for handling incoming requests (e.g., http://+:80/example/).

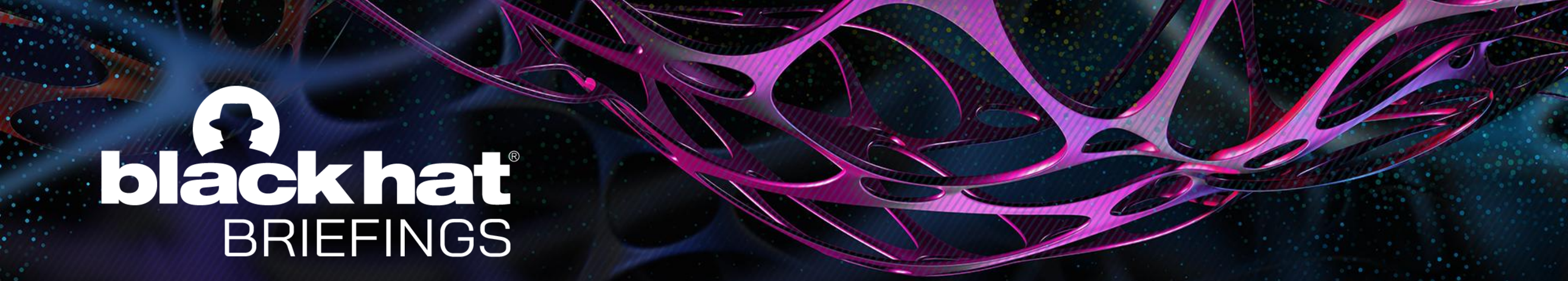
How to find them

- HttpQueryServiceConfiguration
  - A Windows API used to query configuration details managed by HTTP.sys.
  - Can retrieve:
    - Registered URLs
    - SSL certificate bindings
    - IP listeners
    - Request queue names
    - Service SID bindings
  - Allows inspection of system-wide HTTP configuration from user-mode.

How to find them

> netsh http show servicestate

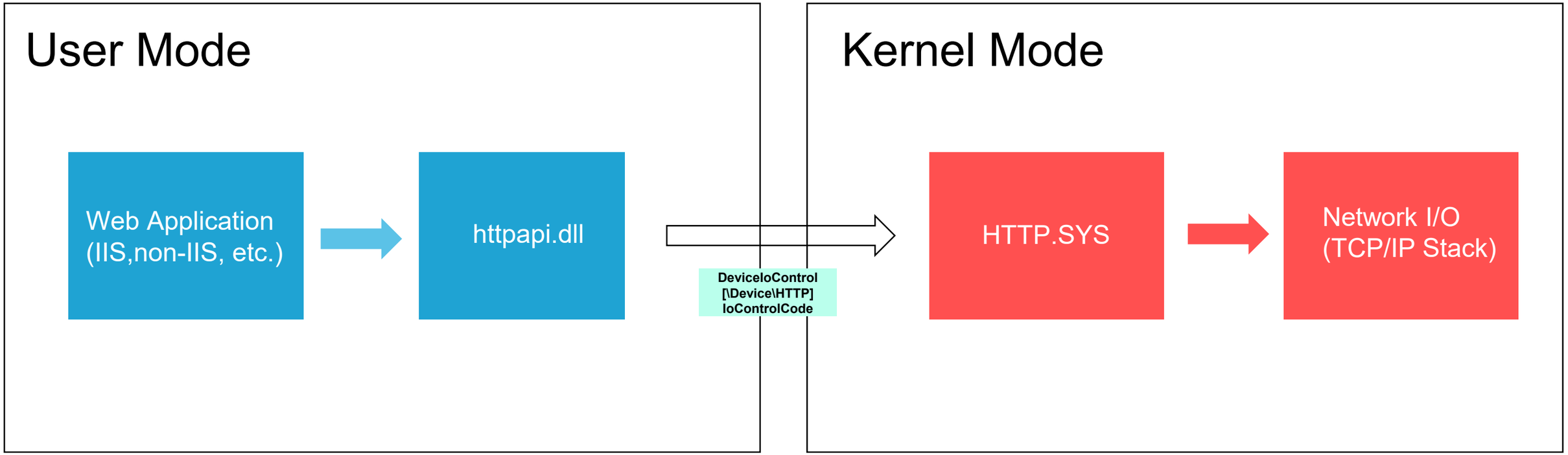
```
Request queues:
    Request queue name: Request queue is unnamed.
        Version: 1.0
        State: Active
        Request queue 503 verbosity level: Basic
        Max requests: 1000
        Active requests: 0
        Queued requests: 0
        Max queued request age: 0s
        Requests arrived: 8
        Requests rejected: 0
        Cache hits: 0
        Number of active processes attached: 1
        Processes:
            ID: 3560, image: C:\Windows\System32\svchost.exe
            Services: WinRM
            Tagged Service: WinRM
        Registered URLs:
            HTTP://+:5985/WSMAN/
            HTTP://+:47001/WSMAN/
```

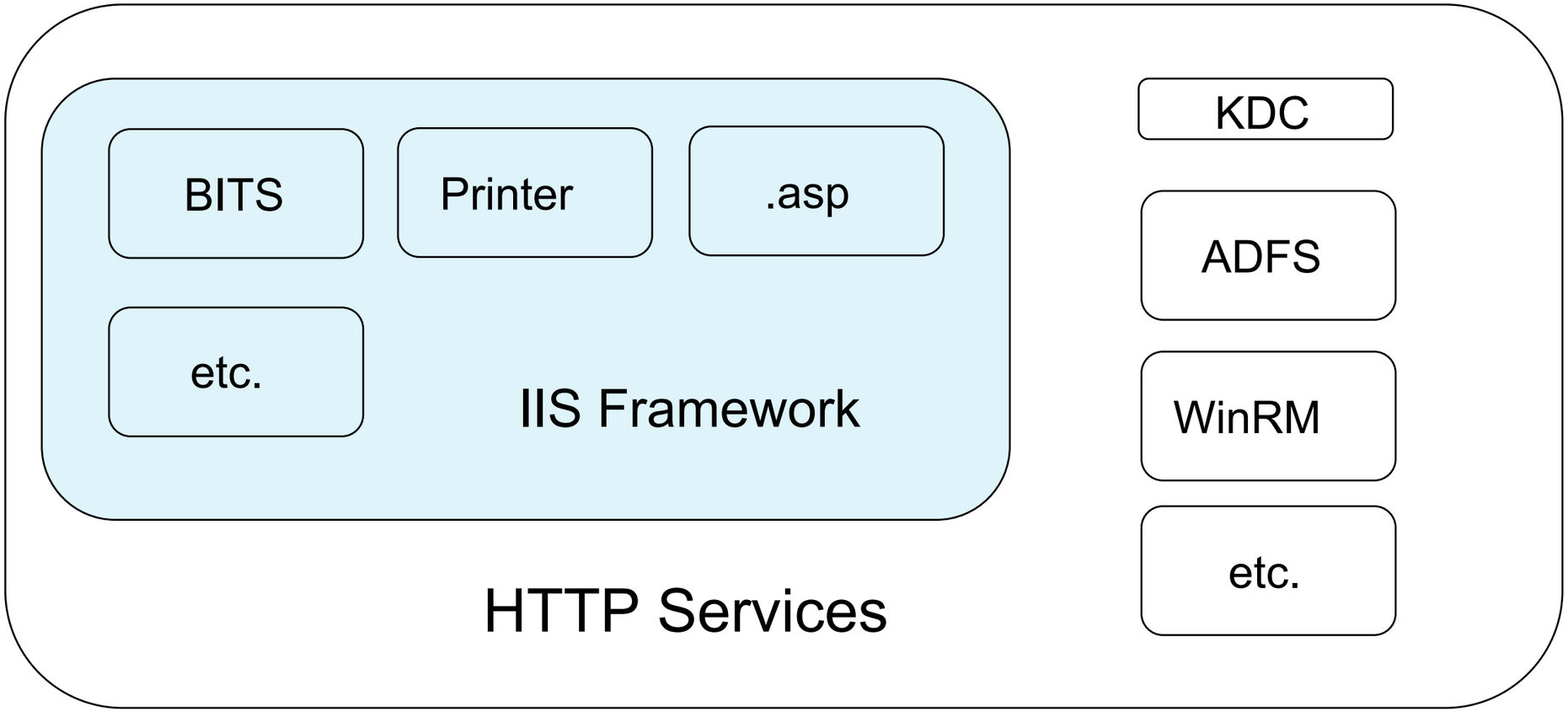


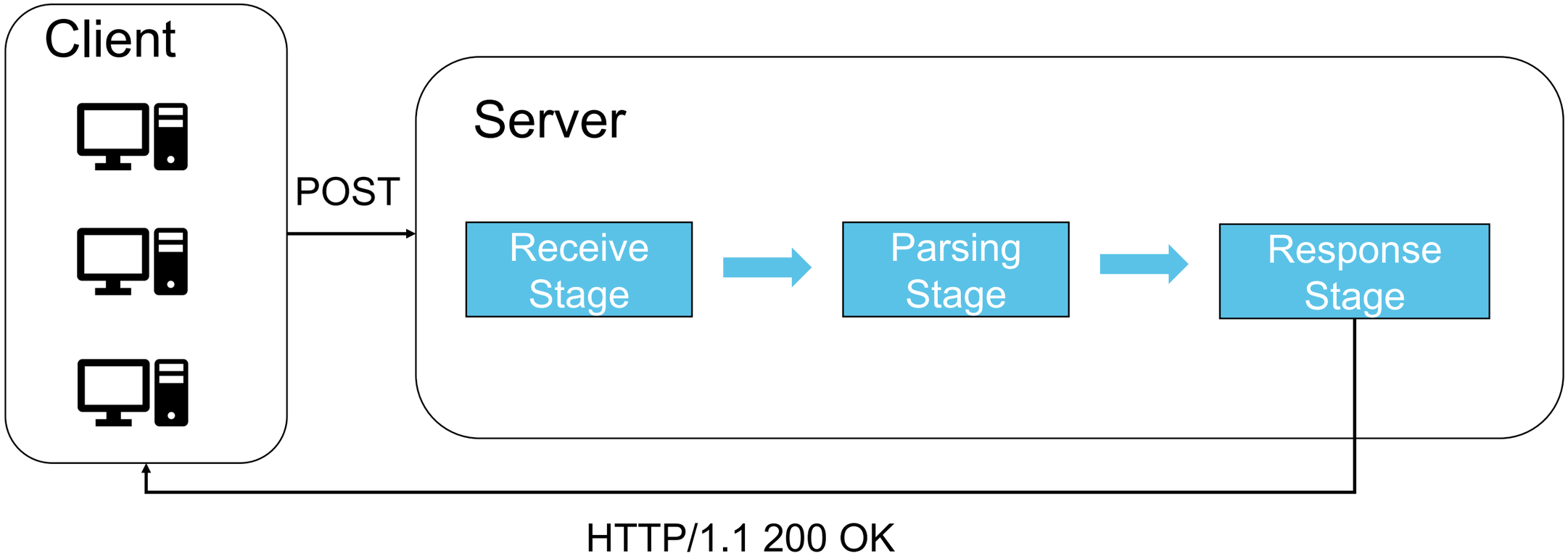
AUGUST 6-7, 2025

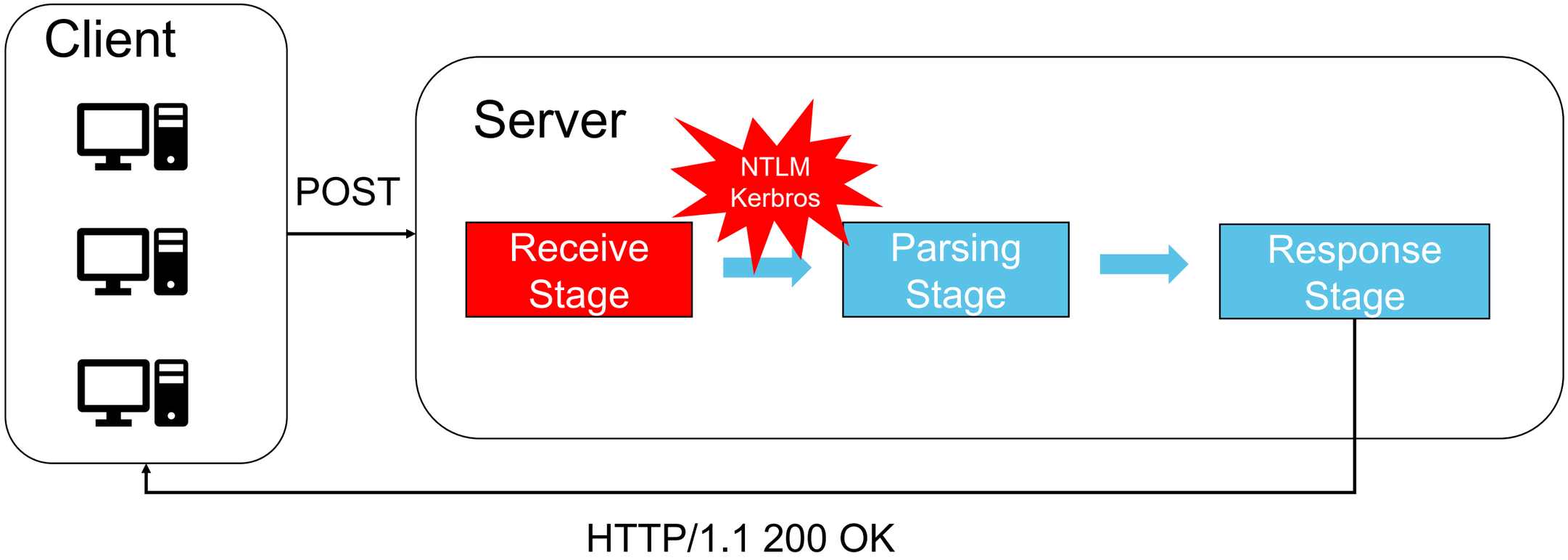
MANDALAY BAY / LAS VEGAS

Overview of the Windows HTTP Service Framework


User Mode
Kernel Mode
Web Application
(IIS,non-IIS, etc.)
httpapi.dll
DeviceIoControl
[\Device\HTTP]
IoControlCode
HTTP.SYS
Network I/O
(TCP/IP Stack)


BITS
Printer
.asp
etc.
IIS Framework
KDC
ADFS
WinRM
etc.
HTTP Services


Client
POST
Server
Receive Stage
Parsing Stage
Response Stage
HTTP/1.1 200 OK


Client
POST
Server
NTLM
Kerbros
Receive Stage
Parsing Stage
Response Stage
HTTP/1.1 200 OK

HTTP Related APIs

HttpReceiveHttpRequest

```
ULONG HttpReceiveHttpRequest(
    [in] HANDLE RequestQueueHandle,
    [in] HTTP_REQUEST_ID RequestId,
    [in] ULONG Flags,
    [out] PHTTP_REQUEST RequestBuffer,
    [in] ULONG RequestBufferLength,
    [out, optional] PULONG BytesReturned,
    [in, optional] LPOVERLAPPED Overlapped
);
```

https://learn.microsoft.com/en-us/windows/win32/api/http/nf-http-httpreceivehttprequest

```
POST /api/example HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0) Gecko/20100101 Firefox/128.0
Content-Type: application/octet-stream
Content-Length: 256
Connection: keep-alive

\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42
\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43
\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44
\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45
\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46
\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47
\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48
\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49
\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A
\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B
\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C
\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D
\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E
\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F
\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50
```

HTTP Related APIs

HttpReceiveRequestEntityBody

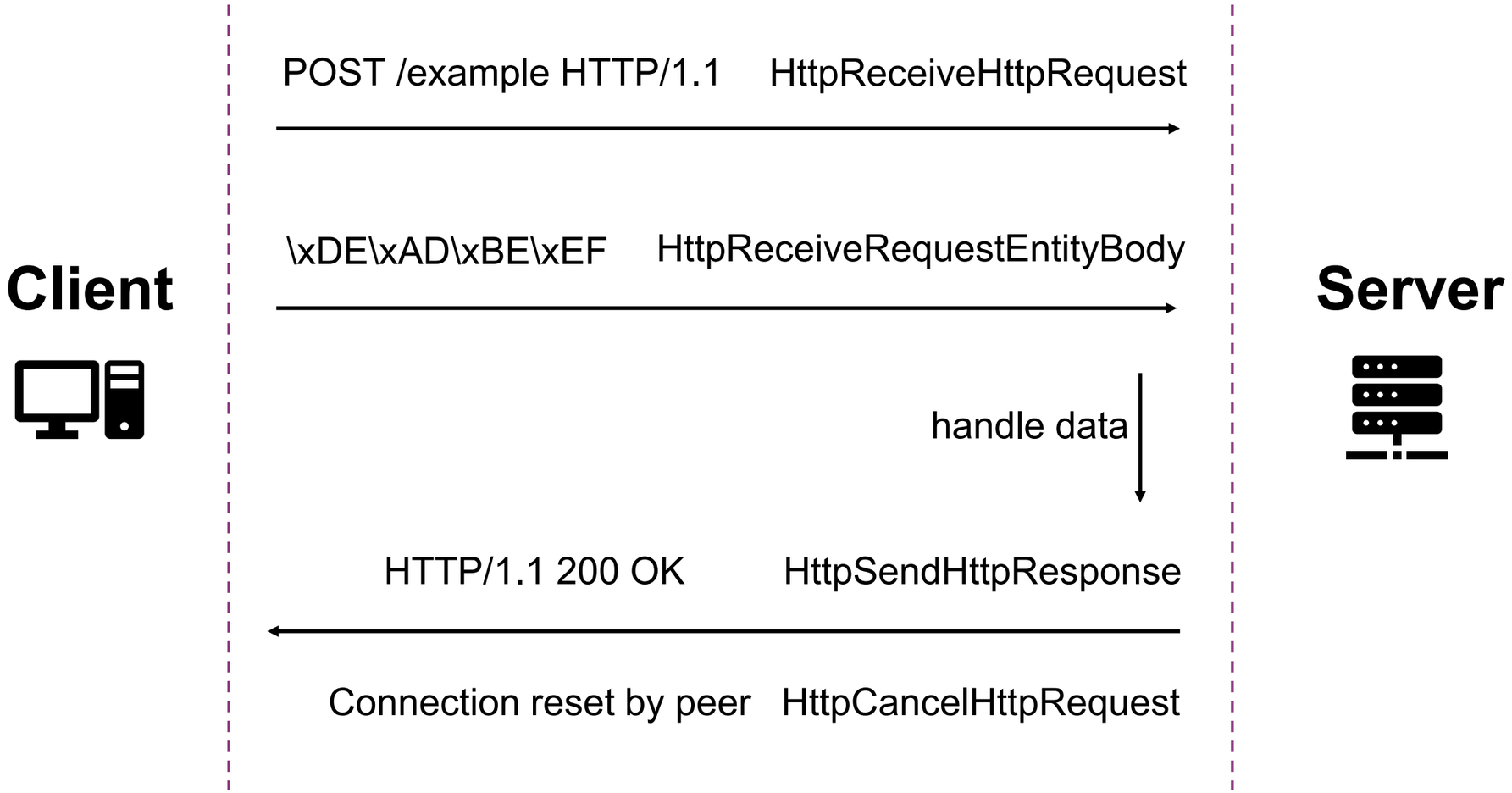
```
ULONG HttpReceiveRequestEntityBody(
    [in] HANDLE RequestQueueHandle,
    [in] HTTP_REQUEST_ID RequestId,
    [in] ULONG Flags,
    [out] PVOID EntityBuffer,
    [in] ULONG EntityBufferLength,
    [out, optional] PULONG BytesReturned,
    [in, optional] LPOVERLAPPED Overlapped
);
```

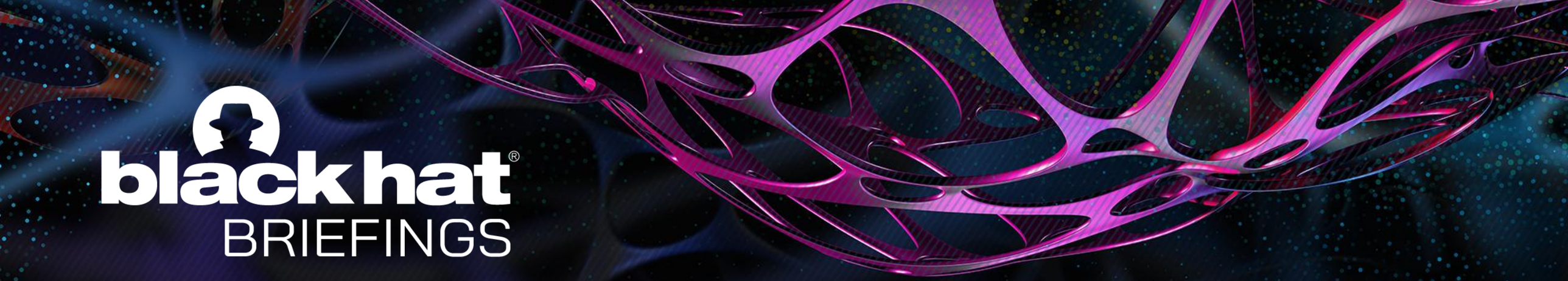
```
POST /api/example HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0) Gecko/20100101 Firefox/128.0
Content-Type: application/octet-stream
Content-Length: 256
Connection: keep-alive

\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41
\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42\x42
\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43\x43
\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44
\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45\x45
\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46\x46
\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47\x47
\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48\x48
\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49
\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A\x4A
\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B\x4B
\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C\x4C
\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D\x4D
\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E\x4E
\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F\x4F
\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50
```

https://learn.microsoft.com/en-us/windows/win32/api/http/nf-http-httpreceiverequestentitybody

HTTP Related APIs





AUGUST 6-7, 2025

MANDALAY BAY / LAS VEGAS

Exploring Logic Flaws Leading to Pre-auth DoS

Tips:

When hunting for pre-auth DoS bugs, it's not only about memory corruption (e.g., null pointer dereference or out-of-bounds read without info leak) — that's just one class of DoS.

What if the server can no longer process any normal requests at all?
That’s DoS too — and sometimes even more impactful.

Considering the framework of Windows HTTP services, I focused on the **receive stage** and the **response stage**, where such logic flaws are most likely to exist.

Receiving Stage

Three Mechanisms:

- Synchronous
- Asynchronous — WaitForMultipleObjects
- Asynchronous — Callback

Synchronous

- Single-threaded
- Service doesn't invoke HttpReceiveHttpRequest until the current request handling finishes

```
void SyncHandleFunction()
{
    […]
    while ( 1 )
    {
            v7 = HttpReceiveHttpRequest(…); // receive http header
            […] // process http header/ POST data / …
    }
    […]
    return;
}
```

Case Study – CVE-2024-43512

Windows Standards-based Storage Management Service

```
bool __fastcall concrete::HTTPListener::Run(HANDLE *this) // concrete.dll
{
  [...]
  LABEL_3:
  while ( 1 )
  {
    while ( 1 )
    {
      memset_0(v3, 0, 0x1360ui64);
      BytesReturned = 0;
      v6 = HttpReceiveHttpRequest(this[1], RequestId, 0, v5, 0x1360u, &BytesReturned, 0i64); // ============> [a]
      if ( v6 == 0xEA ) // ================> [b]
      {
        RequestId = v5->RequestId;
        v2 = (struct _HTTP_REQUEST_V2 *)realloc(v3, BytesReturned); // ================> [c]
        v3 = v2;
        if ( !v2 )
          return 0;
        goto LABEL_3; // ==============> [d]
      }
  [...]
}
```

Never update

Case Study – CVE-2024-43512

Before

```
D:\>python request.py 192.168.217.244
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
<MESSAGE ID="" PROTOCOLVERSION="1.0">
<SIMPLEEXPRSP>
<EXPMETHODRESPONSE NAME="ExportIndication">
<IRETURNVALUE>
</IRETURNVALUE>
</EXPMETHODRESPONSE>
</SIMPLEEXPRSP>
</MESSAGE>
</CIM>
```

After

```
D:\>python request.py 192.168.217.244
Traceback (most recent call last):
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py", line 791, in urlopen
    response = self._make_request(
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py", line 537, in _make_request
    response = conn.getresponse()
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connection.py", line 461, in getresponse
    httplib_response = super().getresponse()
  File "C:\Program Files\Python310\lib\http\client.py", line 1374, in getresponse
    response.begin()
  File "C:\Program Files\Python310\lib\http\client.py", line 318, in begin
    version, status, reason = self._read_status()
  File "C:\Program Files\Python310\lib\http\client.py", line 287, in _read_status
    raise RemoteDisconnected("Remote end closed connection without"
http.client.RemoteDisconnected: Remote end closed connection without response
```

Asynchronous — WaitForMultipleObjects

- Single thread
- Does not block inside HTTP API functions, but waits for a completion signal
- Creates a separate thread to handle the request

```
void AsyncHandleObjectFunction()
{
  […]
  while ( 1 )
  {
        v7 = HttpReceiveHttpRequest(…);  // return 0x3E5, will not block
        […]
        if ( WaitForMultipleObjectsEx(…) != 1 ) // wait for receive http header, set signal
        […]
        if ( GetOverlappedResult() ) //  get return value and overlapped buffer
        […] // process http header/ POST data / … in separate thread
  }
  […]
  return;
}
```

Case Study -- CVE-2025-27471

```
__int64 __fastcall BaseHttpListener::DoReceiveRequestHeaders(BaseHttpListener *this) // upnphost.dll
{
    NumberOfBytesTransferred = 0; // ======================> [a]
    [...]
    LastError = (*(__int64 (__fastcall)())this+11) // HttpReceiveHttpRequest , ==========> [b]
                (
                *((_QWORD *)this + 14),
                RequestId,
                0i64,
                v2,
                v4,
                0i64, // ==================> [c]
                &Overlapped);
    [...]
    case 0xEAu: // ===============> [d]
          v4 = NumberOfBytesTransferred; // ==============>[e]   v4 will always remain at 0 if NumberOfBytesTransferred was not updated
          *((_DWORD *)this + 72) = 0;
          RequestId = v2->RequestId;
          free(v3);
          v2 = (struct _HTTP_REQUEST_V2 *)malloc(v4);
    [...]
    if ( GetOverlappedResult(*((HANDLE *)this + 14), &Overlapped, &NumberOfBytesTransferred, 0) ) // ================> [f]
    [...]
}
```

Case Study -- CVE-2025-27471

upnphost!BaseHttpListener::DoReceiveRequestHeaders+0x166 "r eax;g;"

```
0:001> ba e1 upnphost!BaseHttpListener::DoReceiveRequestHeaders+0x166 "r eax;g;"
0:001> g
Breakpoint 0 hit
httpapi!HttpReceiveHttpRequest:
00007ffc`2eea2910 4053            push    rbx
0:001> bd 0
0:001> g
eax=3e5
eax=0
eax=ea
eax=ea
eax=ea
eax=ea
eax=ea
eax=ea
```

return 0x3e5 or 0x0 as normal

Causes DoS by entering an infinite loop.

Case Study -- CVE-2025-27471

Before

```
D:\>python upnp_normal.py 192.168.217.150
Content-Length: 31
Content-Type: text/html
Server: Microsoft-Windows/10.0 UPnP/1.0 UPnP-Device-Host/1.0 Microsoft-HTTPAPI/2.0
Date: Tue, 01 Jul 2025 05:29:40 GMT
```

After

```
D:\>python upnp_normal.py 192.168.217.150
Exception in thread Thread-1 (thr):
Traceback (most recent call last):
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py", line 791, in urlopen
    response = self._make_request(
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py", line 537, in _make_request
    response = conn.getresponse()
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connection.py", line 461, in getresponse
    httplib_response = super().getresponse()
  File "C:\Program Files\Python310\lib\http\client.py", line 1374, in getresponse
    response.begin()
  File "C:\Program Files\Python310\lib\http\client.py", line 318, in begin
    version, status, reason = self._read_status()
  File "C:\Program Files\Python310\lib\http\client.py", line 279, in _read_status
    line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
  File "C:\Program Files\Python310\lib\socket.py", line 705, in readinto
    return self._sock.recv_into(b)
```

Asynchronous — Callback

- The most popular mechanism in HTTP services.
  Examples: Kerberos Proxy, RDP, RDG, WinRM, ADFS, and even IIS

- Uses a thread pool to create handler threads; each thread handles one request.

  Examples: CreateThreadpoolIo/StartThreadpoolIo/CancelThreadpoolIo

- Registers callbacks to manage every interaction and event.

Asynchronous — Callback

Common Callback Functions

- HandleReceiveRequestIoCompletionCallback
- HandleReceiveEntityIoCompletionCallback
- HandleSendResponseIoCompletionCallback
- HandleCancelResponseIoCompletionCallback

Optional Callbacks (Registered When Needed)

- HandleWaitForDisconnectionIoCompletionCallback
- HttpReceiveClientCertIoCompletionCallback

```
void AsyncHandleIoCompletionRoutine()
{
  […]
  switch ( *((_DWORD *)Overlapped + 8) ) // Depends on the set with each service
  {
    case 1:
      HttpReceiveRequestIoCompletion(IoResult, NumberOfBytesTransferred, Overlapped);
      break;
    case 2:
      HttpSendResponseIoCompletion(IoResult, NumberOfBytesTransferred, Overlapped);
      break;
    case 3:
      HttpSendPostResponseIoCompletion(IoResult, NumberOfBytesTransferred, Overlapped);
      break;
    case 4:
      HttpReceiveRequestEntityIoCompletion(IoResult, NumberOfBytesTransferred, Overlapped);
      break;
    case 5:
      HttpCancelRequestIoCompletion(IoResult, NumberOfBytesTransferred, Overlapped);
      break;[…]
  return;
}
```

Tips:

• In **single-threaded scenarios** (both sync and async), after processing a request, the service calls HttpReceiveHttpRequest again to wait for the next one.

• In the **callback-based model**, the callback function must call StartThreadpoolIo and then invoke HttpReceiveHttpRequest to start a new thread from the IO thread pool for handling the next request.

Think about this situation:
If the callback returns **without** calling HttpReceiveHttpRequest, the current thread will exit. Eventually, if **all threads** in the IO thread pool exit this way, there will be **no handler threads left**, and the service will **never process normal requests again**.

Case Study — WSDApi.dll

```
__int64 __fastcall CWSDHttpListener::HandleRequest( // wsdapi.dll
        CWSDHttpListener *this,
        struct _HttpAsyncRequest *a2,
        __int64 a3,
        int a4)
{
    [...]
    ioresult = *((_DWORD *)a2 + 15); // ==============> [a]
    if ( ioresult )
    {
        Transport = *((_DWORD *)a2 + 15);
        if ( ioresult > 0 )
        Transport = (unsigned __int16)ioresult | 0x80070000; // =================> [b]
        // forget to call HttpReceiveHttpRequest, no http handler anymore
    }
    else
    {
        [...]
        Transport = CWSDHttpListener::IssueReceiveRequest(this, v20, v21, v22); // ==============> [c]
        [...]
    }
    return Transport; // ==============> [d]
}
```

CWSDHttpListener::IoCompletionRoutine

CWSDHttpListener::HandleRequest

Case Study

Before

```
D:\>python fdres.py 192.168.217.150
Content-Type: application/soap+xml
Server: Microsoft-HTTPAPI/2.0
Date: Tue, 01 Jul 2025 05:30:39 GMT
Content-Length: 0
```

After

```
D:\>python fdres.py 192.168.217.150
Traceback (most recent call last):
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py", line 791, in urlopen
    response = self._make_request(
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connectionpool.py", line 537, in _make_request
    response = conn.getresponse()
  File "C:\Users\k0shl\AppData\Roaming\Python\Python310\site-packages\urllib3\connection.py", line 461, in getresponse
    httplib_response = super().getresponse()
  File "C:\Program Files\Python310\lib\http\client.py", line 1374, in getresponse
    response.begin()
  File "C:\Program Files\Python310\lib\http\client.py", line 318, in begin
    version, status, reason = self._read_status()
  File "C:\Program Files\Python310\lib\http\client.py", line 279, in _read_status
    line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
  File "C:\Program Files\Python310\lib\socket.py", line 705, in readinto
    return self._sock.recv_into(b)
```

Case Study – Low severity

MSRC acknowledged it as a pre-auth DoS, but rated it as low severity because the service is only exposed in trusted networks.

Receiving Stage

Impact:

- Unauthenticated: No authentication or extra configuration required
- Easy attack: Triggered by just one or a few malicious packets
- Persistent DoS: Service permanently stops handling requests from legitimate clients, and it doesn't require keep connections from attack client

Secure Development Considerations:

- ✓ Never exit the handler process early; always invoke HttpReceiveHttpRequest with RequestID = 0 to start listening for new requests
- ✓ Pay special attention to error returns, especially 0xEA and 0x4CD
- ✓ Carefully handle variables and states after error returns to avoid inconsistent behavior

Response stage

HttpSendHttpResponse

```
HTTPAPI_LINKAGE ULONG HttpSendHttpResponse(
  [in] HANDLE RequestQueueHandle,
  [in] HTTP_REQUEST_ID RequestId,
  [in] ULONG Flags,
  [in] PHTTP_RESPONSE HttpResponse,
  [in, optional] PHTTP_CACHE_POLICY CachePolicy,
  [out] PULONG BytesSent,
  [in] PVOID Reserved1,
  [in] ULONG Reserved2,
  [in] LPOVERLAPPED Overlapped,
  [in, optional] PHTTP_LOG_DATA LogData
);
```

Response stage

HttpCancelHttpRequest

```
HTTPAPI_LINKAGE ULONG HttpCancelHttpRequest(
  [in] HANDLE RequestQueueHandle,
  [in] HTTP_REQUEST_ID RequestId,
  [in, optional] LPOVERLAPPED Overlapped
);
```

Response Stage

Http.sys – Establishes HTTP Connections on the Server Side

UxTlAllocateConnectionForLookaside

```
ULONGLONG *__fastcall UxTlAllocateConnectionForLookaside(
        POOL_TYPE PoolType,
        SIZE_T NumberOfBytes,
        ULONG Tag,
        PLOOKASIDE_LIST_EX Lookaside)
{
  ULONGLONG Alignment; // rbx
  unsigned int v5; // eax
  __int64 Pool3; // rax
  ULONGLONG *v7; // rdi
  ULONG_PTR v9; // rdx
  int v10; // eax
  KSPIN_LOCK *v11; // rsi
  KIRQL v12; // dl
  unsigned int v13; // ecx
  ULONGLONG v14; // rbx
  ULONGLONG *v15; // rcx
  ULONGLONG **v16; // rax

  Alignment = Lookaside[1].L.ListHead.Alignment;
  v5 = 0x102;
  if ( PoolType != PagedPool )
    v5 = 0x42;
  Pool3 = ExAllocatePool3(v5, NumberOfBytes, 1129606229i64, &UxLowPriorityPool, 1);// allocate nonpaged pool
  v7 = (ULONGLONG *)Pool3;
  if ( !Pool3 )
    return 0i64;
  v9 = Pool3 + 36;
  *(_DWORD *)(Pool3 + 32) = 1129598069;
  *(_DWORD *)(Pool3 + 36) = 1;
  v10 = _InterlockedDecrement((volatile signed __int32 *)(Pool3 + 36));
  if ( UxDebugCheckRefcount && v10 <= -1 )
    UlBugCheckEx(3ui64, v9, 0x21ui64, v10);
  v11 = (KSPIN_LOCK *)(Alignment + 8640);
  v12 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(Alignment + 8640));
  if ( *(_BYTE *)(Alignment + 8648) || (v13 = *(_DWORD *)(Alignment + 8400), v13 >= *(_DWORD *)(Alignment + 8404)) )// check max connection
  {
    KeReleaseSpinLock((PKSPIN_LOCK)(Alignment + 8640), v12);
    ExFreePoolWithTag(v7, 0);
    return 0i64;
  }
  *(_DWORD *)(Alignment + 8400) = v13 + 1;      // increase ref count
  v14 = Alignment + 8408;
  v15 = v7 + 2;
  v16 = *(ULONGLONG ***)(v14 + 8);
  if ( *v16 != (ULONGLONG *)v14 )
    __fastfail(3u);
  *v15 = v14;
  v7[3] = (ULONGLONG)v16;
  *v16 = v15;
  *(_QWORD *)(v14 + 8) = v15;
  KeReleaseSpinLock(v11, v12);
  return v7;
}
```

Default Maximum Connections

- Default value: 0xFFFFFFFF (unlimited)
- Can be configured via the registry:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\HTTP\Parameters
Registry value: MaxConnections

Response Stage

Http.sys – After Disconnection

- Triggered by actions like HttpSendHttpResponse or HttpCancelHttpRequest

UxTlFreeConnectionFromLookaside

```
1  void __fastcall UxTlFreeConnectionFromLookaside(_QWORD *Buffer, PLOOKASIDE_LIST_EX Lookaside)
2  {
3    ULONGLONG Alignment; // rbx
4    KIRQL v4; // al
5    _QWORD *v5; // rdx
6    __int64 v6; // r9
7    _QWORD *v7; // r8
8
9    Alignment = Lookaside[1].L.ListHead.Alignment;
10   v4 = KeAcquireSpinLockRaiseToDpc((PKSPIN_LOCK)(Alignment + 8640));
11   v5 = Buffer + 2;
12   v6 = Buffer[2];
13   if ( *(_QWORD **)(v6 + 8) != Buffer + 2 || (v7 = (_QWORD *)Buffer[3], (_QWORD *)*v7 != v5) )
14     __fastfail(3u);
15   *v7 = v6;
16   *(_QWORD *)(v6 + 8) = v7;
17   *v5 = 0i64;
18   Buffer[3] = 0i64;
19   --*(_DWORD *)(Alignment + 8400);              // decrease ref count
20   KeReleaseSpinLock((PKSPIN_LOCK)(Alignment + 8640), v4);
21   ExFreePoolWithTag(Buffer, 0);                 // free nonpaged pool
22 }
```

Response Stage

So what happens if the server ends the handler without calling HttpSendHttpResponse or HttpCancelHttpRequest?

Response Stage

So what happens if the server ends the handler without calling HttpSendHttpResponse or HttpCancelHttpRequest?

Connection Resource Leak

- ✓ Connection reference count **never decreases**
- ✓ Connection-related structures **are never freed** from nonpaged pool
- ✓ Causes **nonpaged pool memory exhaustion** over time

Response Stage

BranchCache

- Refer to [MS-PCCRR](https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-pccrr/6409c168-8a3a-473c-b333-6438f067ef56)
- Specific POST Data format

MSG_GETBLKLIST

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	9	30	1
SizeOfSegmentID																															
SegmentID (variable)																															
...																															
ZeroPad (variable)																															
...																															
NeededBlocksRangeCount																															
NeededBlockRanges (variable)																															
...																															

Case Study -- CVE-2024-38149

BranchCach -- CTnoDownloadMgr::OnMessage

```
__int64 __fastcall CTnoDownloadMgr::OnMessage(
        char a1,
        unsigned int a2,
        unsigned int a3,
        void *a4,
        _QWORD *a5,
        __int64 a6,
        int a7,
        __int64 a8)
{
[...]
    switch ( a3 ) // ==============> [a]
    {
 case 1u: // ==============> [b]
        v19 = 1;
        if ( v20 != (TraceLoggingHProvider)&WPP_GLOBAL_Control
          && (*((_BYTE *)v20 + 108) & 8) != 0
          && *((_BYTE *)v20 + 105) >= 4u )
        {
          WPP_SF_qqq(*((_QWORD *)v20 + 12), 49i64, &WPP_152a8e42b8b337334125d2feda130716_Traceguids, a4, *v14, v14[1]);
          goto LABEL_50;
        }
        break;
[...]
    CTnoDownloadMgr::LogInvalidMessage(a6 + 8, v19, 1002i64);
    SystemError::ThrowHelper(L"CTnoDownloadMgr::OnMessage", -2147024122); // =============> [c]
[...]
}
```

[a] Variable a3 can be controlled via POST data
[b] When a3 == 1, it represents MSG_NEGO_REQ
[c] This can trigger exceptions in the service

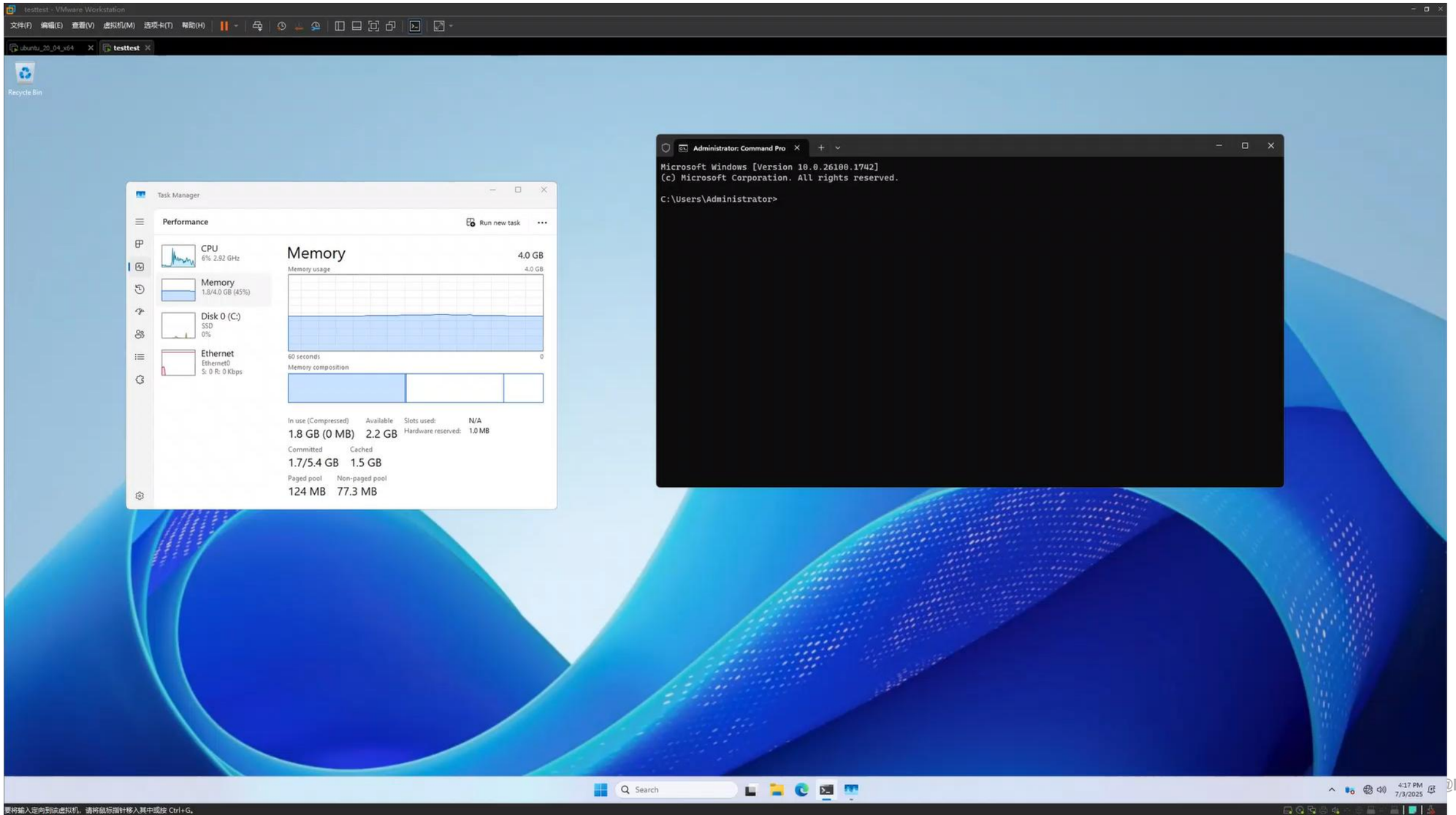
•All POST message types have exception handlers, but malformed data can cause exceptions

•After exception, service **does NOT** call HttpSendHttpResponse or HttpCancelHttpRequest to disconnect

•If attacker **does NOT** disconnect either → nonpaged pool memory leaks

•Leads to **kernel nonpaged pool exhaustion → denial of service**

Case Study -- CVE-2024-38149



Response stage

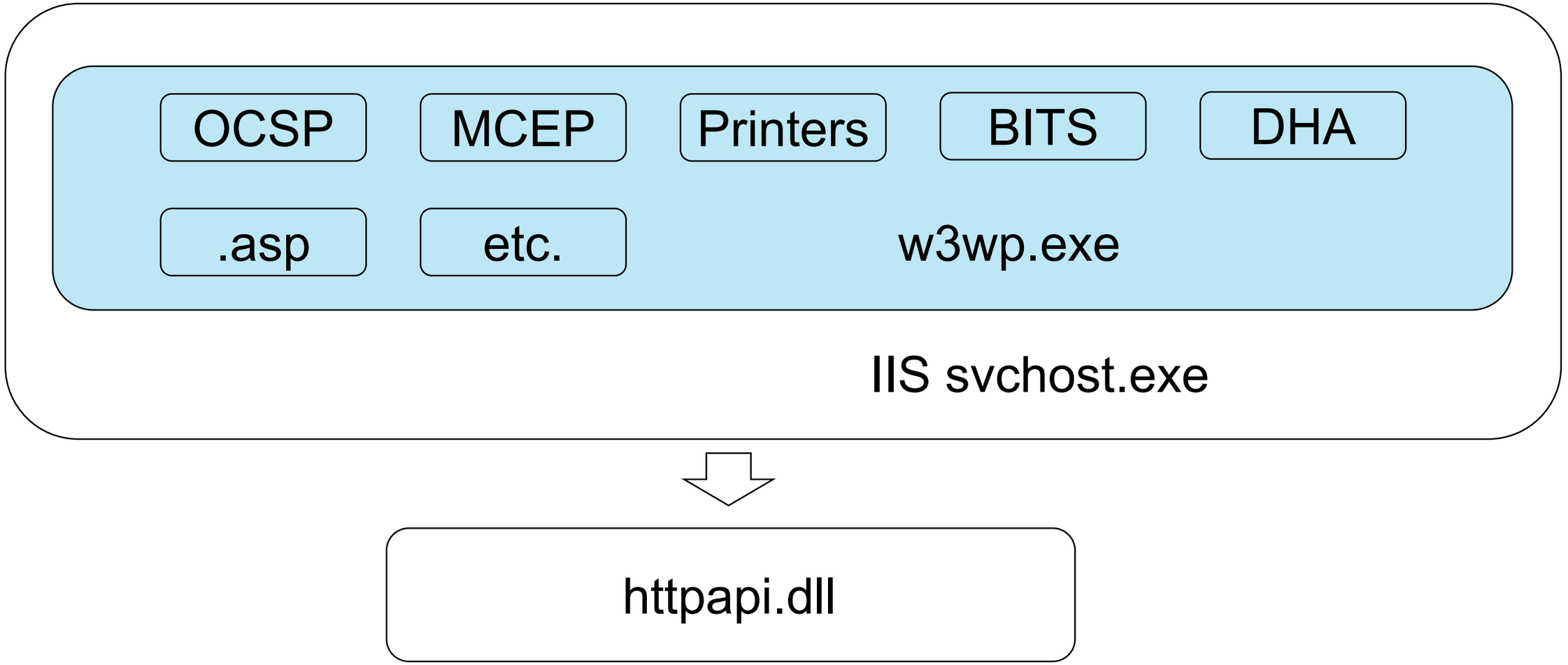
Impact:

- Unauthenticated: No authentication or additional configuration needed
- Kernel Crash: Can cause system hang or Blue Screen of Death (BSoD) due to nonpaged pool memory exhaustion

Secure Development Considerations:

✓ Ensure every request handler always ends by sending a response back or canceling the request(or disconnection callback)

IIS


OCSP
MCEP
Printers
BITS
DHA
.asp
etc.
w3wp.exe
IIS svchost.exe
httpapi.dll

IIS

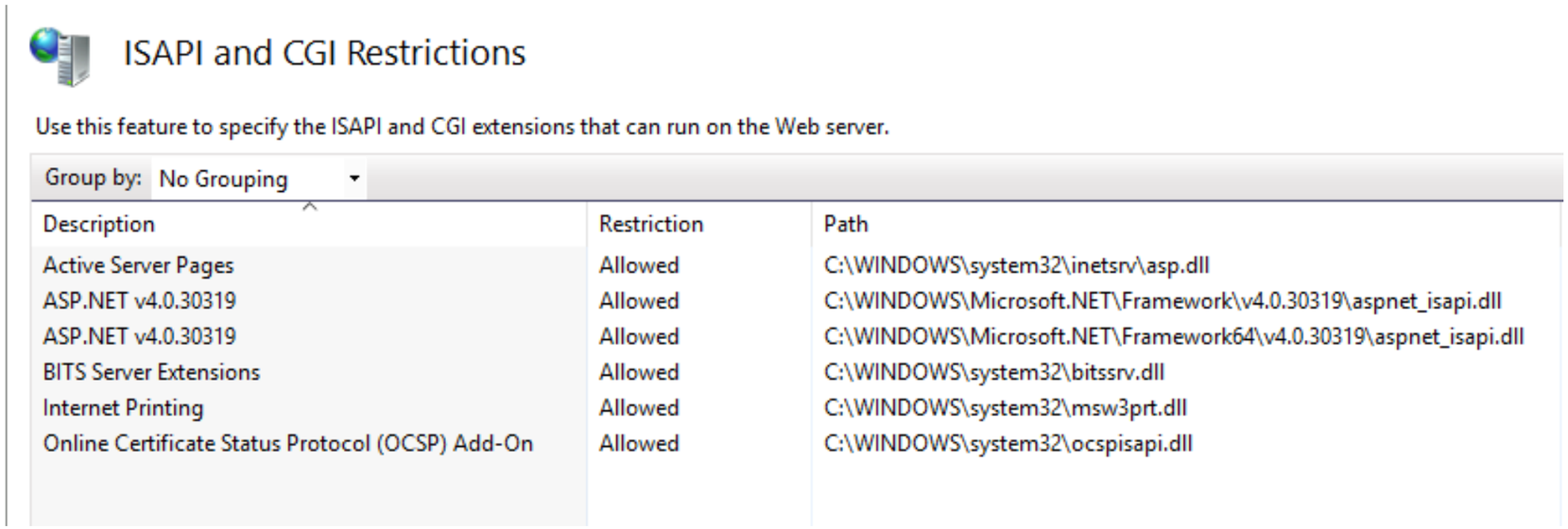
HttpExtensionProc

DWORD WINAPI HttpExtensionProc(LPEXTENSION_CONTROL_BLOCK lpECB);

ISAPI Extensions in IIS

- ✓ Every IIS web server uses ISAPI extensions to process requests
- ✓ Even .asp and C# applications rely on their respective ISAPI extensions
- ✓ For example, servers handling .aspx files use ISAPI extensions like aspnet_isapi.dll or webengine64.dll
- ✓ Although it looks like the web server is processing .aspx directly, the underlying processing is done through ISAPI extensions

IIS



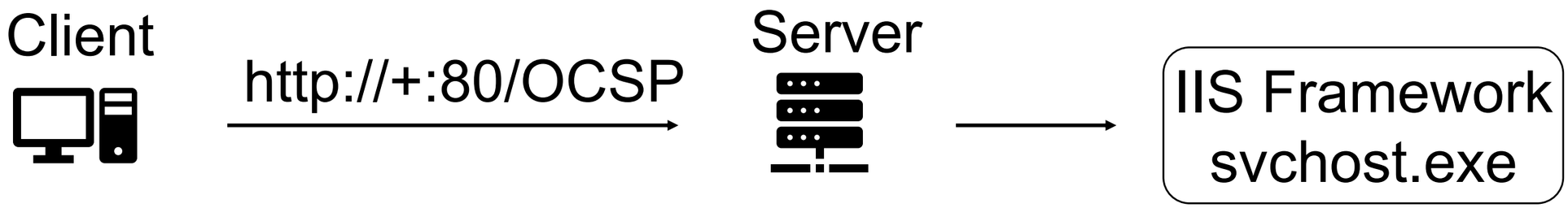
ISAPI and CGI Restrictions

Use this feature to specify the ISAPI and CGI extensions that can run on the Web server.

Group by: No Grouping

Description	Restriction	Path
Active Server Pages	Allowed	C:\WINDOWS\system32\inetsrv\asp.dll
ASP.NET v4.0.30319	Allowed	C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll
ASP.NET v4.0.30319	Allowed	C:\WINDOWS\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.dll
BITS Server Extensions	Allowed	C:\WINDOWS\system32\bitssrv.dll
Internet Printing	Allowed	C:\WINDOWS\system32\msw3prt.dll
Online Certificate Status Protocol (OCSP) Add-On	Allowed	C:\WINDOWS\system32\ocspisapi.dll

IIS




```
appcmd set config "Default Web Site/" /section:system.webServer/handlers /+[name='handler-
wa',path='*',verb='*',modules='IsapiModule',scriptProcessor='"C:\Windows\System32\ocspisapi.dll"',resourceType='
Unspecified',requireAccess='None',preCondition='classicMode']
```


```
    Request queue name: OCSPISAPIAppPool
        Security descriptor: O:BAG:SYD:AI(A;;FR;;;S-1-5-82-1983582526-649149898-3029591049-4059792650-2395287681)(A;;FA;
;;SY)
        Version: 2.0
        State: Active
        Request queue 503 verbosity level: Limited
        Max requests: 1000
        Active requests: 0
        Queued requests: 0
        Max queued request age: 0s
        Requests arrived: 1
        Requests rejected: 0
        Cache hits: 0
        Number of active processes attached: 1
        Controller process:
            ID: 1244, image: C:\Windows\System32\svchost.exe
            Services: WAS, W3SVC
            Tagged Service: WAS
        Processes:
            ID: 2912, image: C:\Windows\System32\inetsrv\w3wp.exe
        Registered URLs:
            HTTP://*:80/OCSP/
```

IIS

ISAPI_CONTEXT Lifecycle in IIS

- For each IIS service, IIS initializes an ISAPI_CONTEXT structure
- For every incoming request:
  - → IIS **increments** the reference count of ISAPI_CONTEXT
  - → After request handling completes, IIS **decrements** the ref count
- When the reference count reaches zero, the structure is released

isapi.dll!ProcessIsapiRequest → ISAPI_CONTEXT:: ISAPI_CONTEXT

iiscore.dll!W3_CONTEXT::SetupStateMachine → Check ref count of ISAPI_CONTEXT

IIS

iiscore.dll! W3_CONTEXT::SetupStateMachine

```
LABEL_41:
    if ( (*(__int64 (__fastcall **)(W3_CONTEXT *))(*(_QWORD *)this + 232i64))(this)
      || (v41 = _InterlockedExchangeAdd((volatile signed __int32 *)(*((_QWORD *)this + 1011) + 212i64), 1u),
          v42 = *((_QWORD *)this + 0x3F3),
          v43 = v41 + 1,
          *((_BYTE *)this + 8097) = 1,
          *(_BYTE *)(v42 + 216))
      && v43 <= *(_DWORD *)(v42 + 208) )
    {
      if ( !(*(__int64 (__fastcall **)(W3_CONTEXT *))(*(_QWORD *)this + 232i64))(this) )
      {
        v44 = (void (__fastcall ***)(__int64, __int64, __int64))(*((_QWORD *)this + 1009) + 656i64);
        v45 = *(_DWORD *)(*(_QWORD *)(*((_QWORD *)this + 6) + 40i64) + 36i64);
        if ( v45 != 4 )
        {
```

if failed

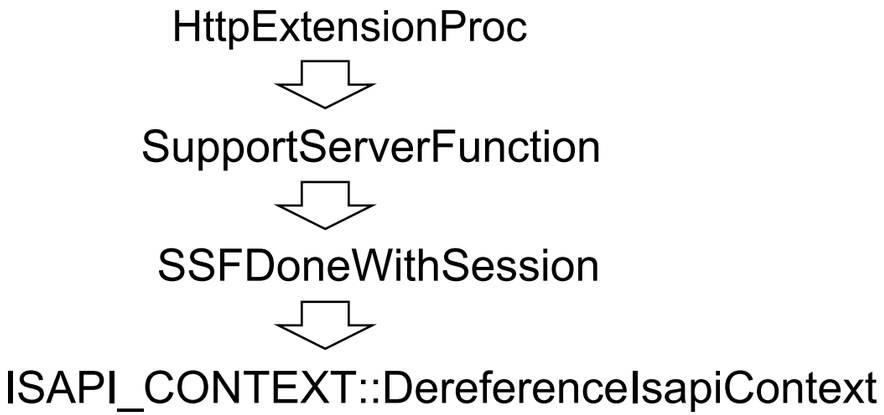
```
v64 = *((_QWORD *)this + 8);
v67 = "Service Unavailable";
v66 = 503;
++*(_DWORD *)(v64 + 624);
v71 = 0;
v65 = 2;
goto LABEL_92;
```

Max ref count is 0x1366

IIS

Balancing ISAPI_CONTEXT Reference Count

- After an ISAPI extension DLL handles data, IIS helps manage ISAPI_CONTEXT reference counts
  - It provides a support function: ServerSupportFunction
- Certain operations in ServerSupportFunction invoke ISAPI_CONTEXT::DereferenceIsapiContext to decrement the reference count
  - Example: SSFDoneWithSession triggers dereference




```
__int64 __fastcall SSFDoneWithSession(struct ISAPI_CONTEXT *this, unsigned int *a2)
{
  __int64 v3; // rsi

  v3 = *((_QWORD *)this + 24);
  if ( (g_dwDebugFlags & 3) != 0 && (g_dwDebugFlags & 0x4000000) != 0 )
    PuDbgPrint(
      g_pDebug,
      "servercommon\\inetsrv\\iis\\iisrearc\\iis70\\isapi\\server_support.cxx",
      933i64,
      "SSFDoneWithSession",
      "\r\n  HSE_REQ_DONE_WITH_SESSION[%p]: Function Entry\r\n  <END>\r\n\r\n",
      this);
  if ( a2 && *a2 == 2 && *((_DWORD *)this + 62) )
  {
    *((_DWORD *)this + 63) = 1;
    (*(void (__fastcall **)(__int64, _QWORD))(*(_QWORD *)v3 + 56i64))(v3, 0i64);
  }
  ISAPI_CONTEXT::DereferenceIsapiContext(this);
  return 0i64;
}
```

IIS

Responsibility of Handling ServerSupportFunction

- ServerSupportFunction is **invoked by the ISAPI extension** via HttpExtensionProc
- It is **not called by IIS**
- This means **each IIS-based service** (e.g., aspnet_isapi.dll, webengine.dll, or any custom extension)
  → must handle it **explicitly and correctly**

IIS

Responsibility of Handling ServerSupportFunction

- ServerSupportFunction is **invoked by the ISAPI extension** via HttpExtensionProc
- It is **not called by IIS**
- This means **each IIS-based service** (e.g., aspnet_isapi.dll, webengine.dll, or any custom extension)
  → must handle it **explicitly and correctly**

Incorrect handling of ServerSupportFunction can silently break the request lifecycle and lead to service-wide impact.

Case Study -- CVE-2024-38067

ocspisapi.dll – httpextensionproc → OcspSvc::COcspIsapiExtension::DispatchStencilCall

```
__int64 __fastcall OcspSvc::COcspIsapiExtension::DispatchStencilCall(
        OcspSvc::COcspIsapiExtension *this,
        struct ATL::AtlServerRequest *a2)
{
[...]
 v34 = OcspSvc::OCSPRequestContext::Decode((OcspSvc::OCSPRequestContext *)&v73, *((_DWORD *)v3 + 366), &v85); // ==============> [a]
   v13 = v34;
[...]
  v38 = OcspSvc::OCSPRequestContext::Validate( // ===============> [b]
          (OcspSvc::OCSPRequestContext *)&v73,
          *((_DWORD *)v3 + 0x16D),
          *((_DWORD *)v3 + 0x16F));
[...]
else if ( !v5
        || (v62 = OcspSvc::COcspIsapiExtension::SendOCSPStatus(v59, v5), (v63 = v62) != 0) // =============> [c]
        && (CSPrintErrorLineFile(0x8AB09C6u, v62), v63 < 0) )
}
```

[a] OCSP server receives an unauthenticated HTTP POST request and decodes the POST data using CryptDecodeObjectEx
[b] The decoded data is then processed by the OCSP service logic
[c] Regardless of success or failure, the server sends an OCSP response back to the client by calling OcspSvc::COcspIsapiExtension::SendOCSPStatus

Case Study -- CVE-2024-38067

ocspisapi.dll – OcspSvc::COcspIsapiExtension::DispatchStencilCall → OcspSvc::COcspIsapiExtension::SendResponseToClient → ServerSupportFunction

```
__int64 __fastcall OcspSvc::COcspIsapiExtension::SendResponseToClient(
        struct ATL::AtlServerRequest *a1,
        struct OcspSvc::COcspResponseCacheEntry *a2,
        char *a3,
        struct _CRYPTOAPI_BLOB *a4,
        int a5)
{
[...]
    if ( (*(unsigned int (__fastcall **)(*((_QWORD *)a1 + 12) + 0xB8i64))( // ============> [d]
            *(_QWORD *)(*((_QWORD *)a1 + 12) + 8i64),
            1037i64,
            v26,
            0i64,
            0i64) )
[...]
}
```

[d] Internally, SendOCSPStatus calls SendResponseToClient, which invokes ServerSupportFunction (IIS dispatch API), eventually reaching W3_RESPONSE::WriteEntityChunks through SSFVectorSend

Case Study -- CVE-2024-38067

ocspisapi.dll – ServerSupportFunction → W3_RESPONSE::WriteEntityChunks

```
signed int __fastcall W3_RESPONSE::WriteEntityChunks(
        W3_RESPONSE *this,
        struct _HTTP_DATA_CHUNK *a2,
        unsigned int a3,
        unsigned int a4,
        int a5,
        struct W3_CONTEXT_BASE *a6,
        unsigned int *a7,
        int *a8)
{
[...]
  if ( v13 )
  {
    result = W3_RESPONSE::Flush(this, a4, a5, a6, a7, a8); // ================> [e]
     if(result ==  error){
          return;
     }
   }
[…]
   W3_CONTEXT_BASE::PostCompletion(*((W3_CONTEXT_BASE **)this + 6), 0, 2); // ===============> [f]
[...]
}
```

[e] After sending the OCSP status back to the client, the server calls PostCompletion
[f] PostCompletion sets the I/O completion callback using PostQueuedCompletionStatus
→ As a result, the session is closed and the ISAPI_CONTEXT structure is dereferenced and eventually released

However, if the unauthenticated client disconnects the TCP connection with the OCSP server before the status is sent back, the **W3_RESPONSE::Flush** function fails and returns a negative error value. As a result, it returns without posting the completion status, and the session will no longer be closed. **The reference count of the ISAPI_CONTEXT will never decrease.** When the reference count reaches 0x1366, the OCSP service will stop receiving requests and return a “503 Service Unavailable" error to any normal client.

Case Study -- CVE-2024-38067

Before

```
D:\>python ocsp_demo.py 192.168.217.244
200
Cache-Control: no-cache
Content-Type: application/ocsp-response
Server: Microsoft-IIS/10.0
Date: Tue, 01 Jul 2025 03:11:27 GMT
Content-Length: 5
```

After

```
D:\>python ocsp_demo.py 192.168.217.244
503 Service Unavailable
```

IIS

Impact:

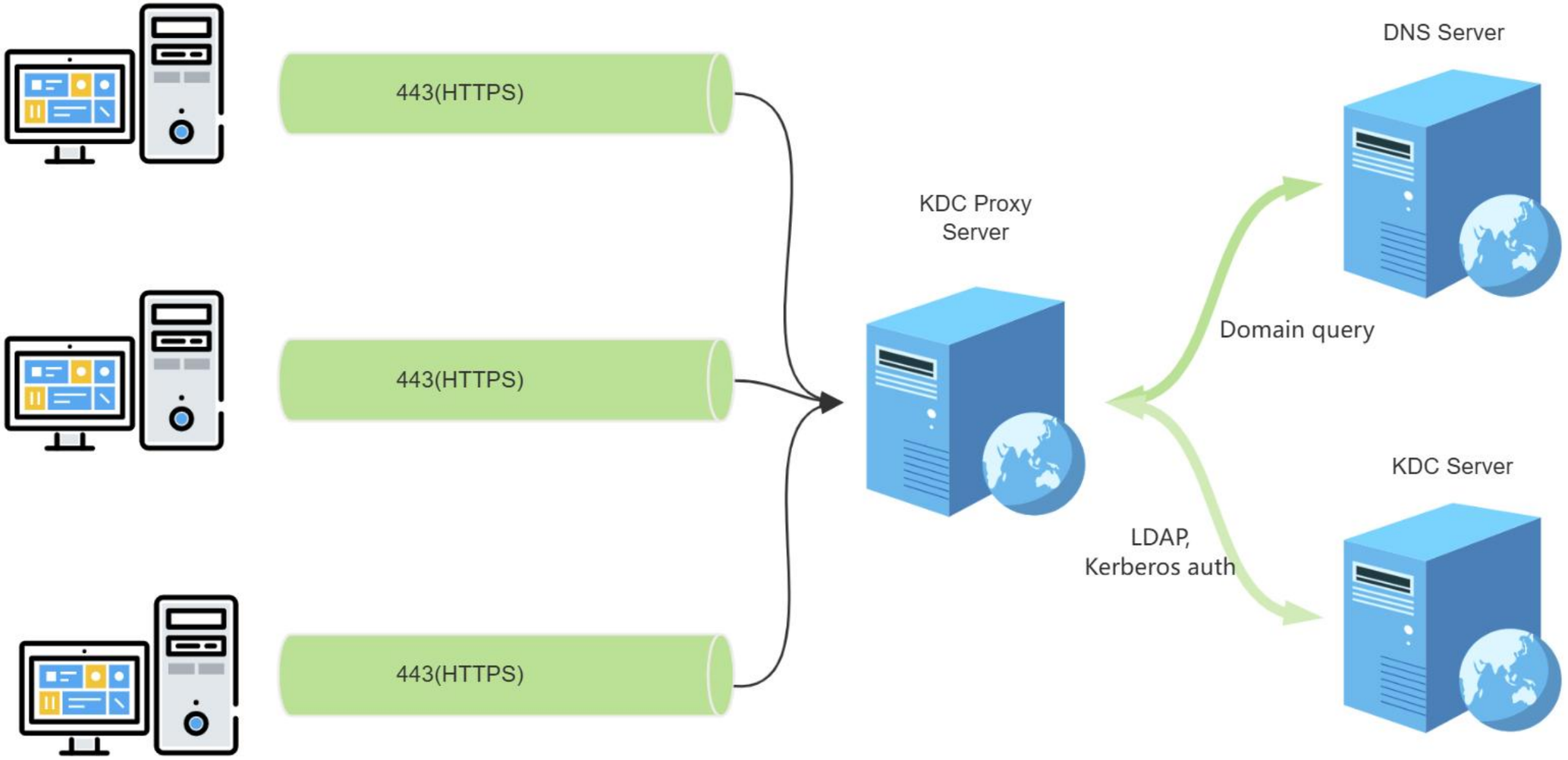
- Unauthenticated: No authentication or extra configuration required
- Persistent DoS: Service permanently stops handling requests from legitimate clients; does not require maintaining connections from the attacker
- More Severe: Mishandling ISAPI_CONTEXT reference counting can not only cause persistent DoS but also lead to use-after-free remote code execution — a common issue when pointer reference counts are mishandled

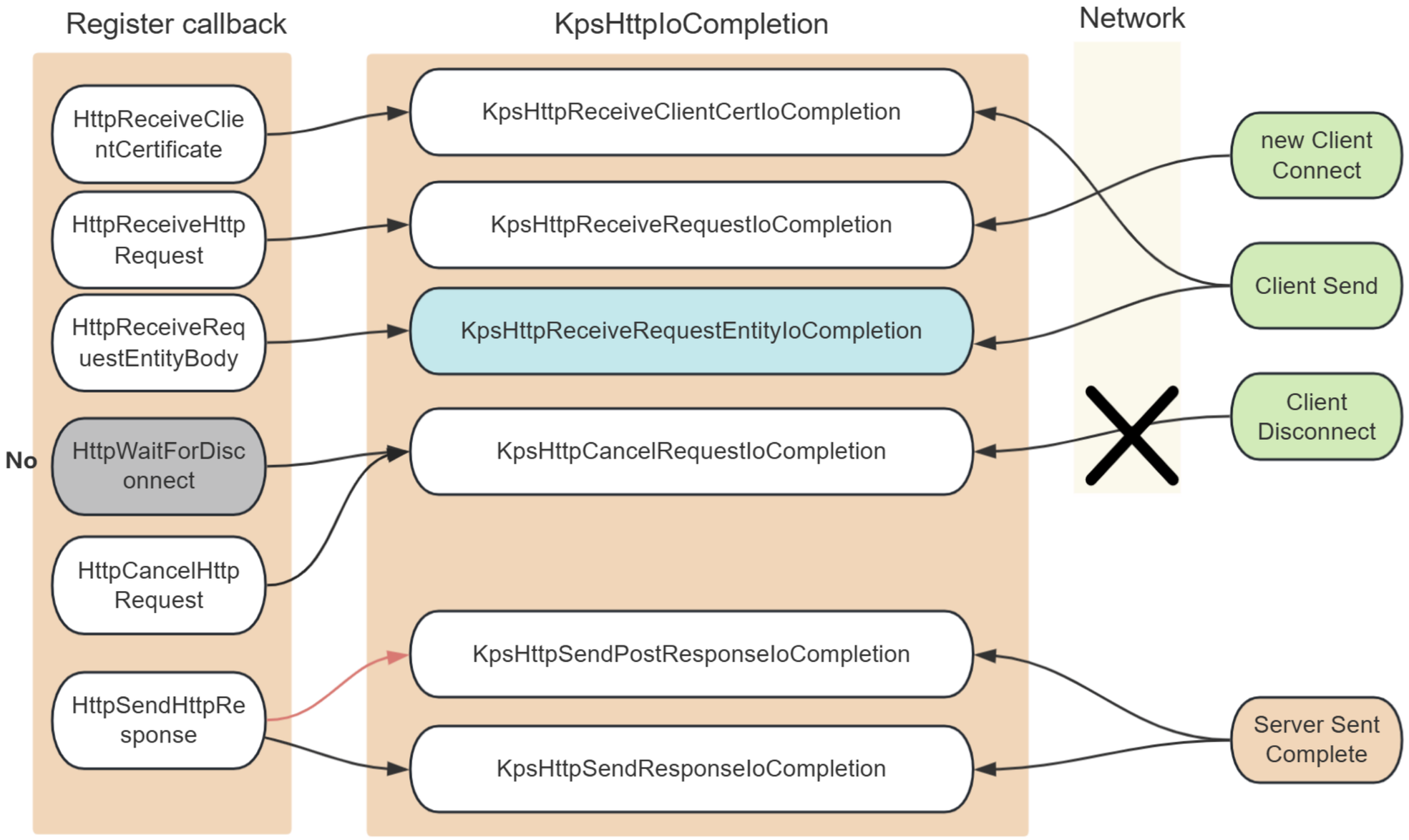
Secure Development Considerations:

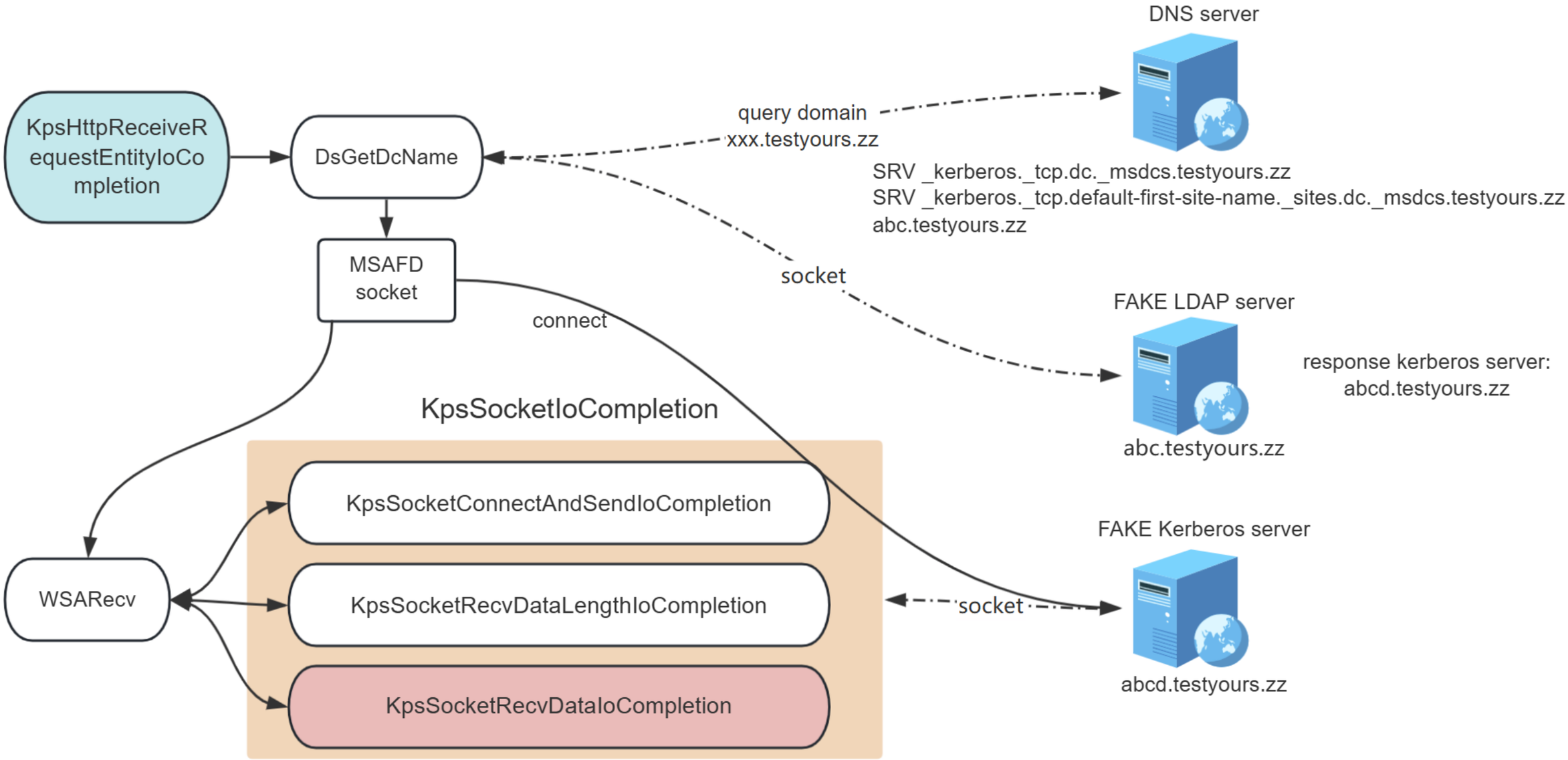
✓ Use ServerSupportFunction carefully within HttpExtensionProc of your ISAPI extension DLL, especially for dispatch routines that manage referencing and dereferencing of ISAPI_CONTEXT.

Parsing and Handling Stages Leading to Pre-auth RCE

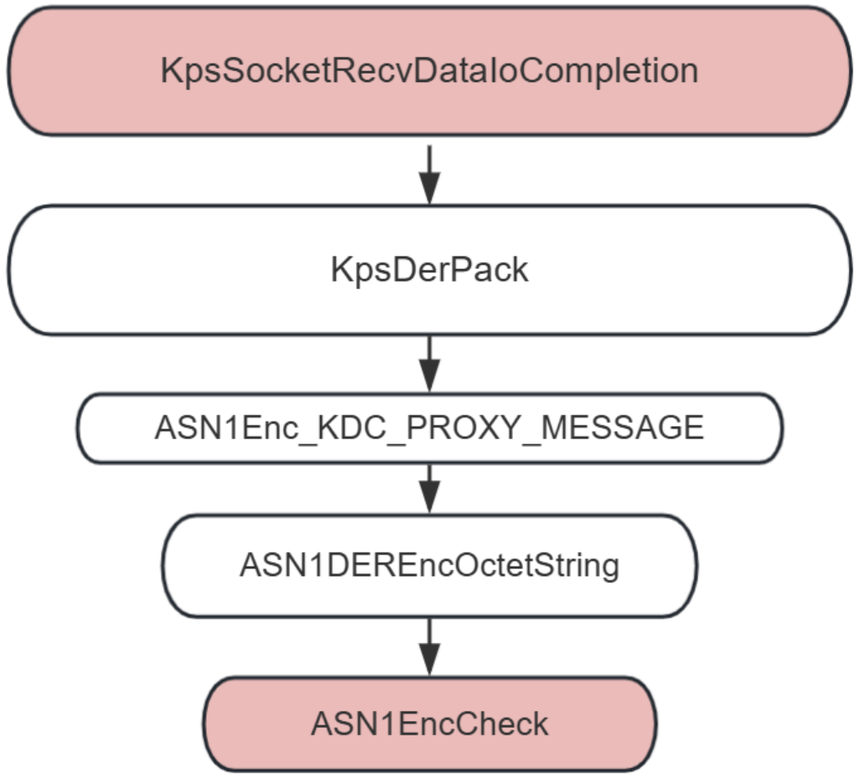
KDC Proxy HTTP Server




Register callback
KpsHttpIoCompletion
Network
HttpReceiveClientCertificate
HttpReceiveHttpRequest
HttpReceiveRequestEntityBody
No
HttpWaitForDisconnect
HttpCancelHttpRequest
HttpSendHttpResponse
KpsHttpReceiveClientCertIoCompletion
KpsHttpReceiveRequestIoCompletion
KpsHttpReceiveRequestEntityIoCompletion
KpsHttpCancelRequestIoCompletion
KpsHttpSendPostResponseIoCompletion
KpsHttpSendResponseIoCompletion
new Client Connect
Client Send
Client Disconnect
Server Sent Complete


DNS server
KpsHttpReceiveRequestEntityIoCompletion
DsGetDcName
query domain
xxx.testyours.zz
SRV _kerberos._tcp.dc._msdcs.testyours.zz
SRV _kerberos._tcp.default-first-site-name._sites.dc._msdcs.testyours.zz
abc.testyours.zz
MSAFD socket
socket
FAKE LDAP server
connect
response kerberos server:
abcd.testyours.zz
abc.testyours.zz
KpsSocketIoCompletion
KpsSocketConnectAndSendIoCompletion
FAKE Kerberos server
WSARecv
KpsSocketRecvDataLengthIoCompletion
socket
KpsSocketRecvDataIoCompletion
abcd.testyours.zz

Case Study -- CVE-2024-43639




```
1   _int64 __fastcall ASN1EncCheck(Encoder* a1, unsigned int a2)
2   {
3     ......
4     dword18 = a1->cur_size_18h;
5     if ( (__int64)pvoid10 + dword18 - a1->cur_buf_28h - (a1->dword24 != 0) >= a2 )
6       return 1;
7     if ( (a1->byte38 & 8) == 0 )
8     {
9       v9 = a1->cur_size_18h;
10      if ( a2 > dword18 )
11        v9 = a2; // 0xfffffffb
12      v10 = dword18 + v9; // 0xfffffffb+5 => 0
13      a1->cur_size_18h = v10;
14      v11 = LocalReAlloc(pvoid10, v10, 0x42u);
```

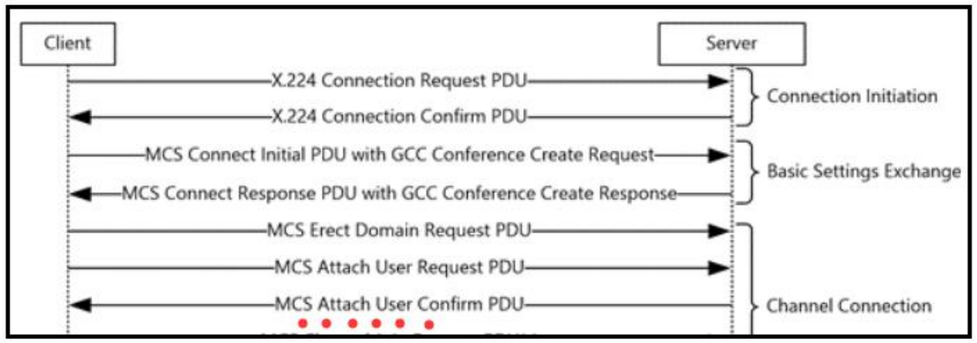
Case Study -- CVE-2024-43639

```
0:007> r
rax=000001af7606b005 rbx=000001af76066fb0 rcx=0000000000000084
rdx=000001af73ac01c0 rsi=00000000fffffbfb rdi=0000000000000005
rip=00007ffd5217740d rsp=0000004a8837f230 rbp=0000000000000000
 r8=7fffffffffffffc  r9=0000004a87f53000 r10=00000fffa9f5a744
r11=4000001000000410 r12=0000000000000000 r13=00007ffd41d50048
r14=0000004a8837f3e8 r15=0000004a8837f3f0
iopl=0         nv up ei pl nz na pe nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
MSASN1!ASN1BEREncLength+0x4d:
00007ffd`5217740d 8808            mov     byte ptr [rax],cl ds:000001af`7606b005=??
```

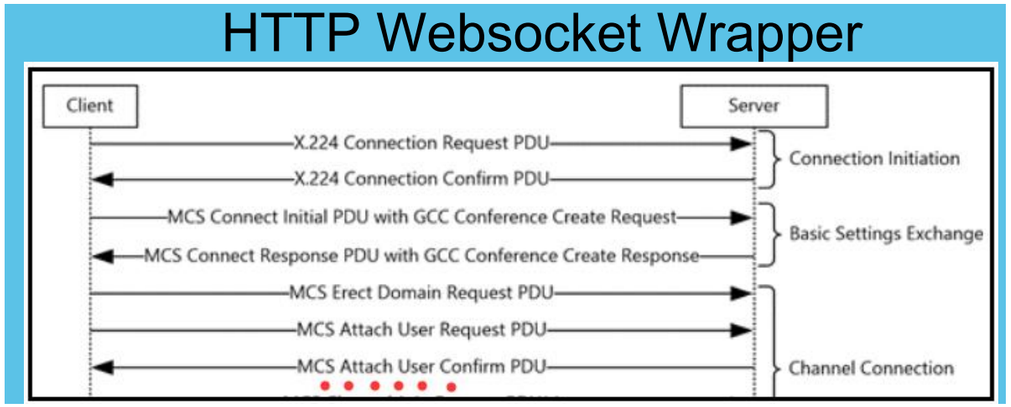
```
0:007> k
 # Child-SP          RetAddr               Call Site
00 0000004a`8837f230 00007ffd`52176a4b     MSASN1!ASN1BEREncLength+0x4d
01 0000004a`8837f260 00007ffd`41d2ea03     MSASN1!ASN1BEREncCharString+0x2b
02 0000004a`8837f290 00007ffd`52177802
kpssvc!ASN1Enc_KDC_PROXY_MESSAGE+0x73
03 0000004a`8837f2d0 00007ffd`41d40900     MSASN1!ASN1_Encode+0xa2
04 0000004a`8837f300 00007ffd`41d42325     kpssvc!KpsDerPack+0xdc
05 0000004a`8837f360 00007ffd`41d3e9e5     kpssvc!KpsPackProxyResponse+0xcd
06 0000004a`8837f3e0 00007ffd`41d3e7a2
kpssvc!KpsSocketRecvDataIoCompletion+0x20d
07 0000004a`8837f460 00007ffd`52f01f31     kpssvc!KpsSocketIoCompletion+0xb2
```

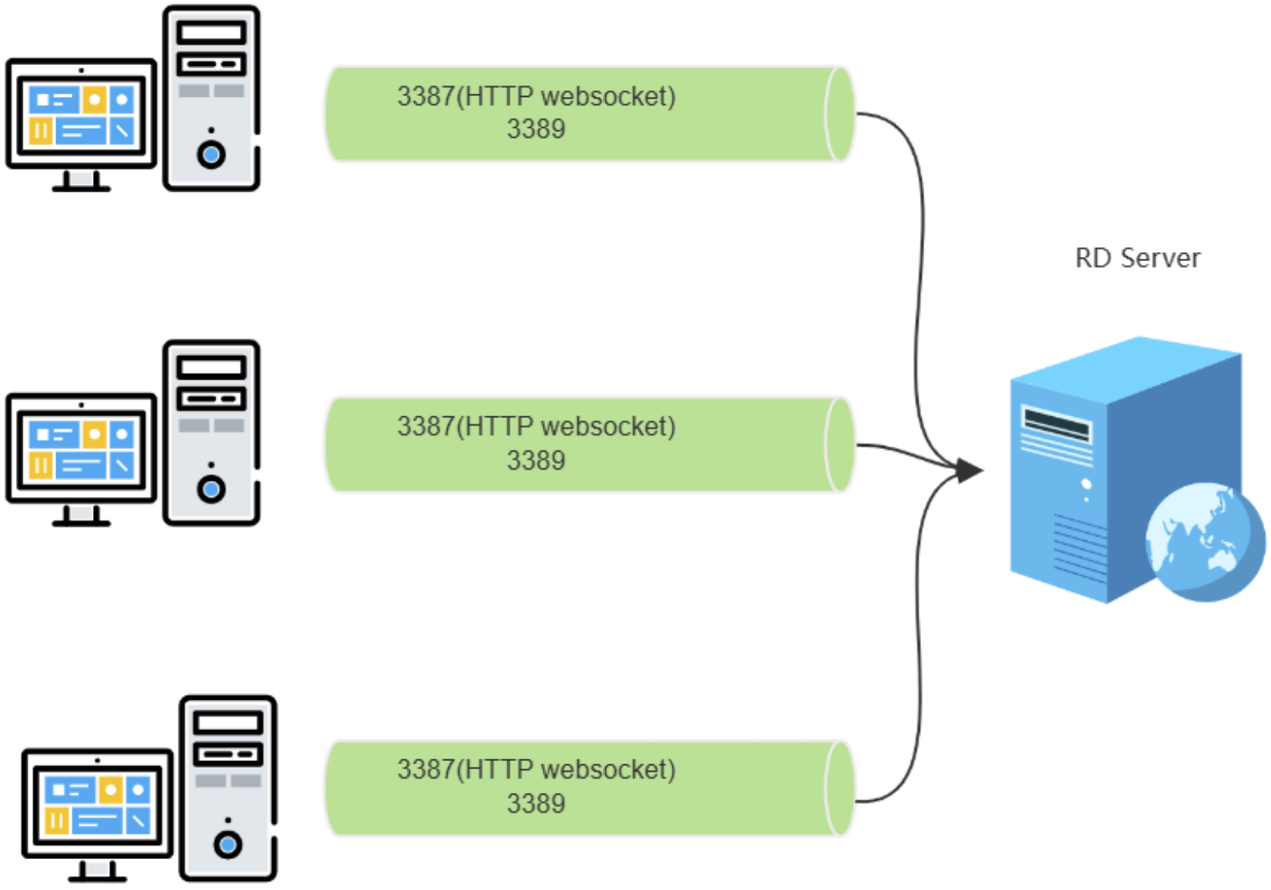
Remote Desktop Service

3389:

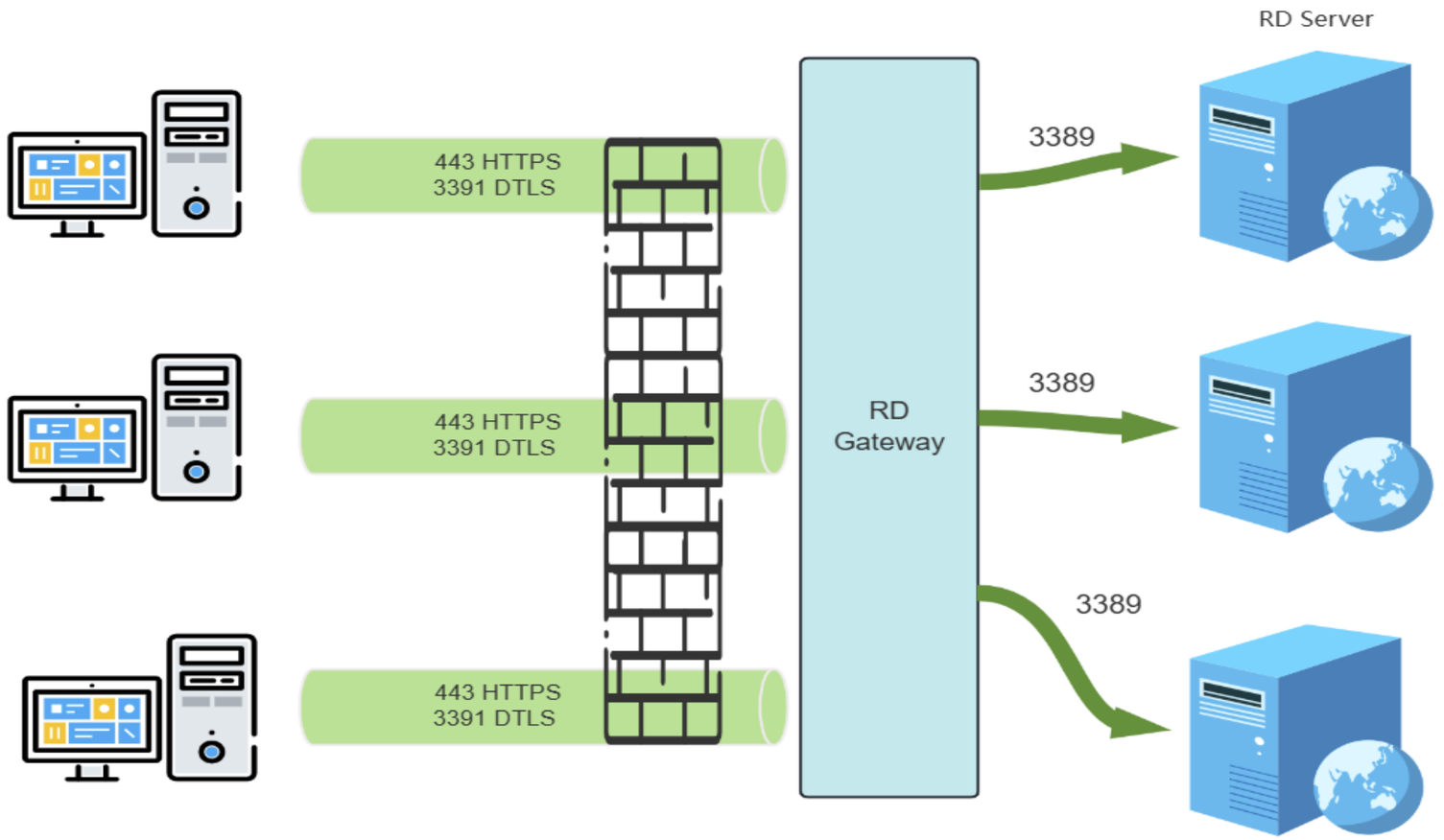


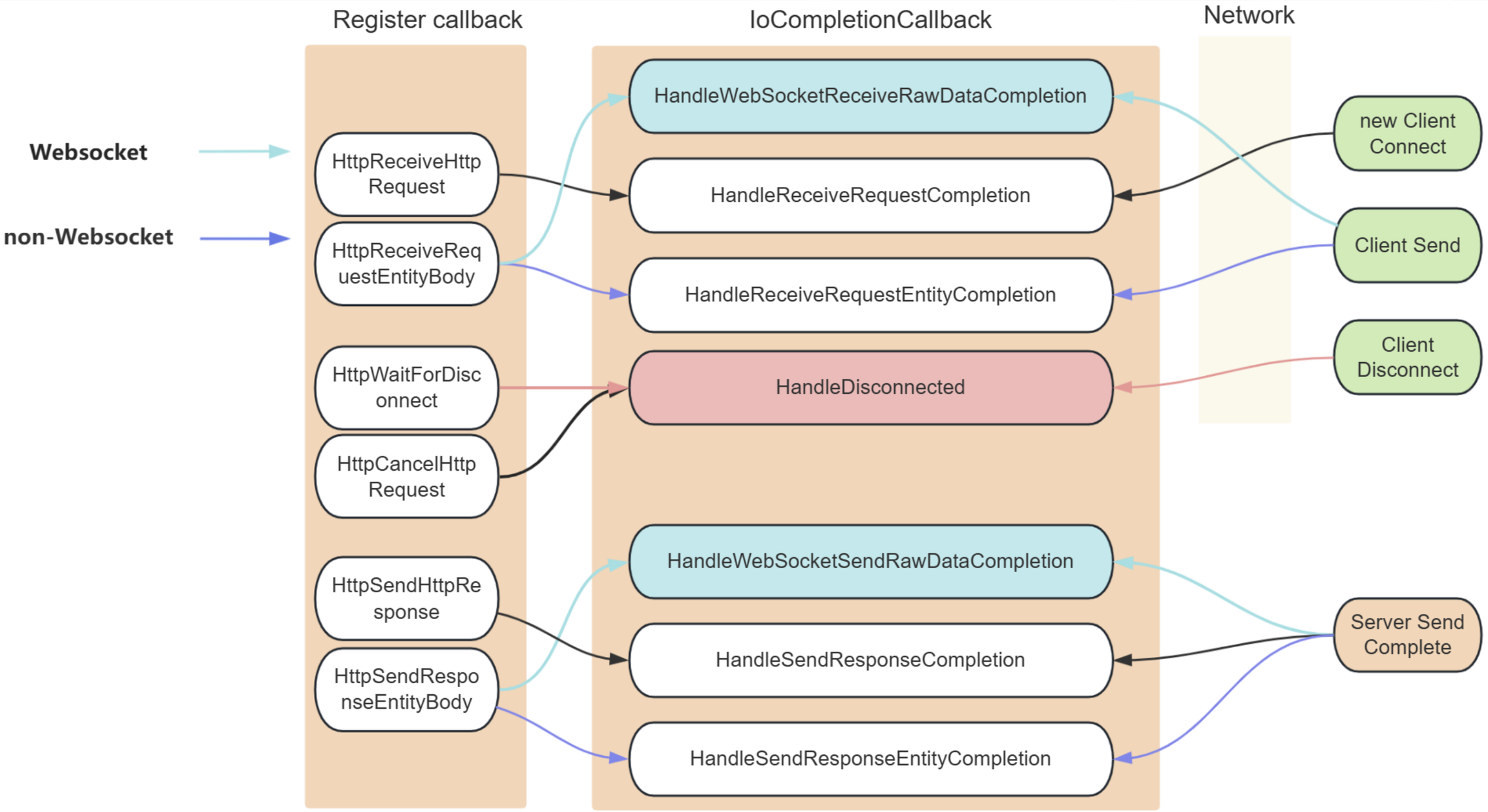
3387:

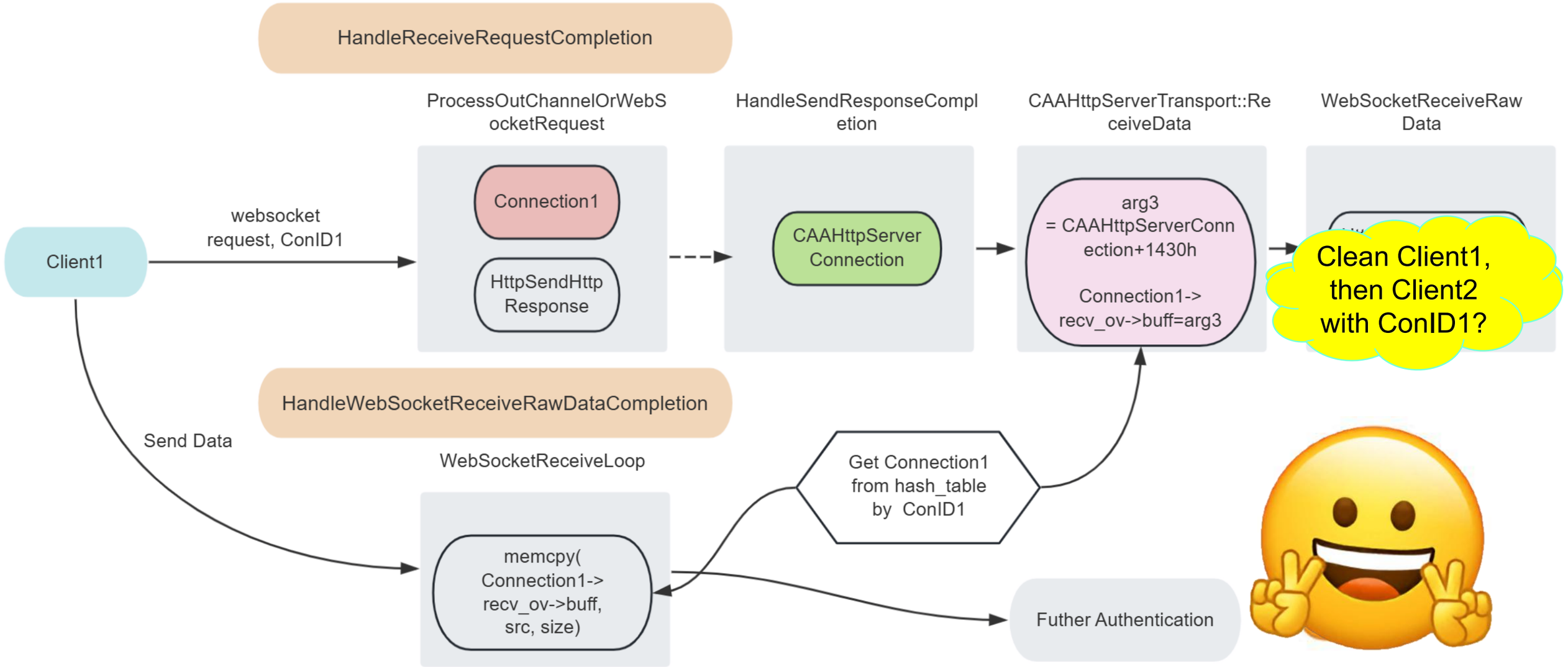




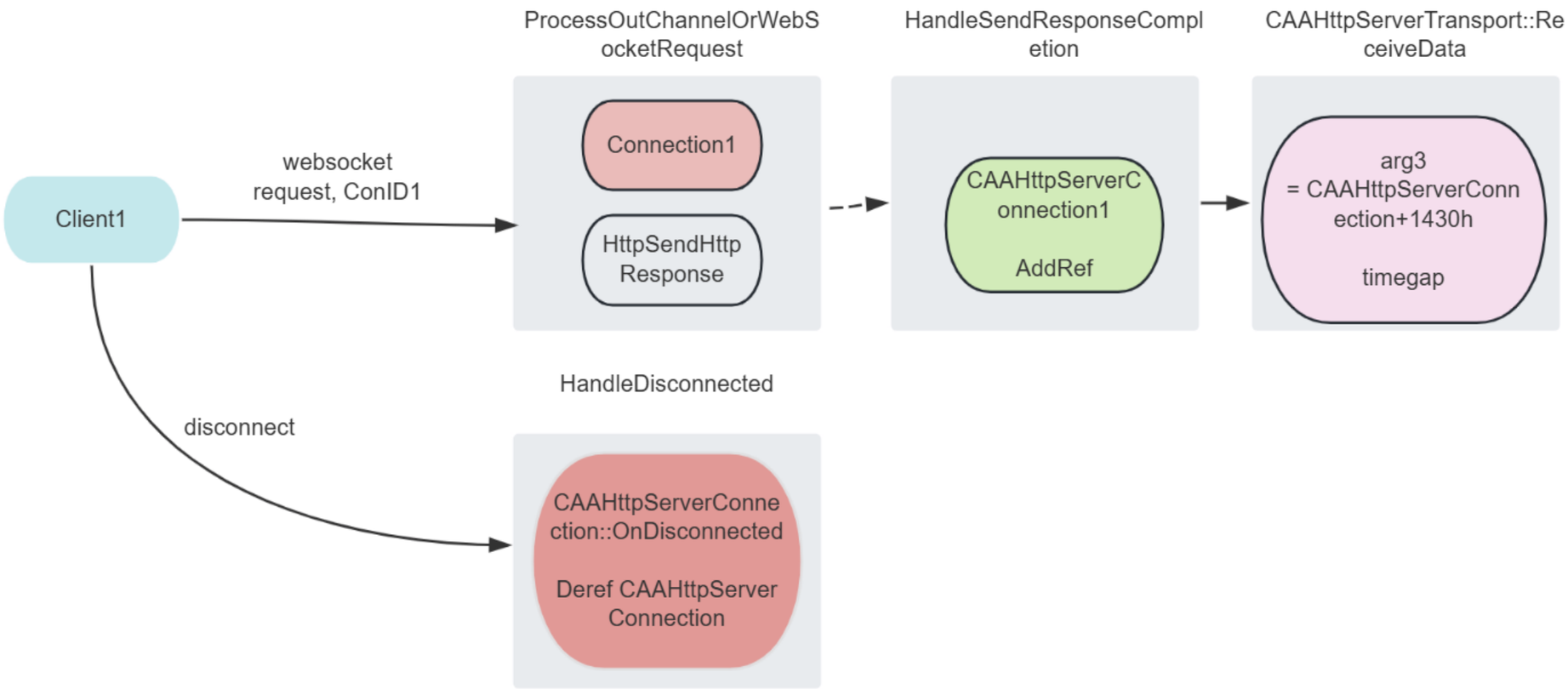
Remote Desktop Gateway Service

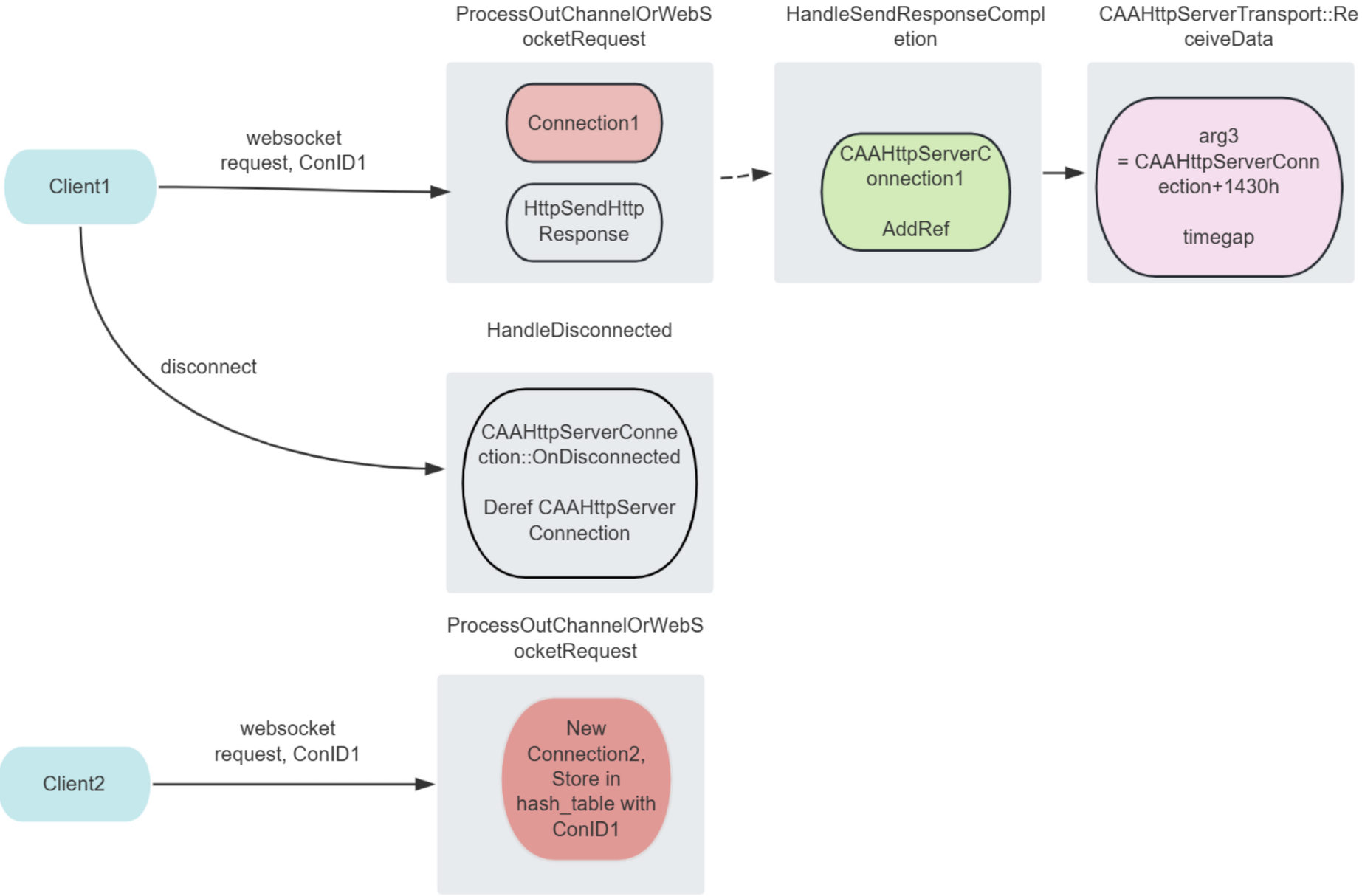


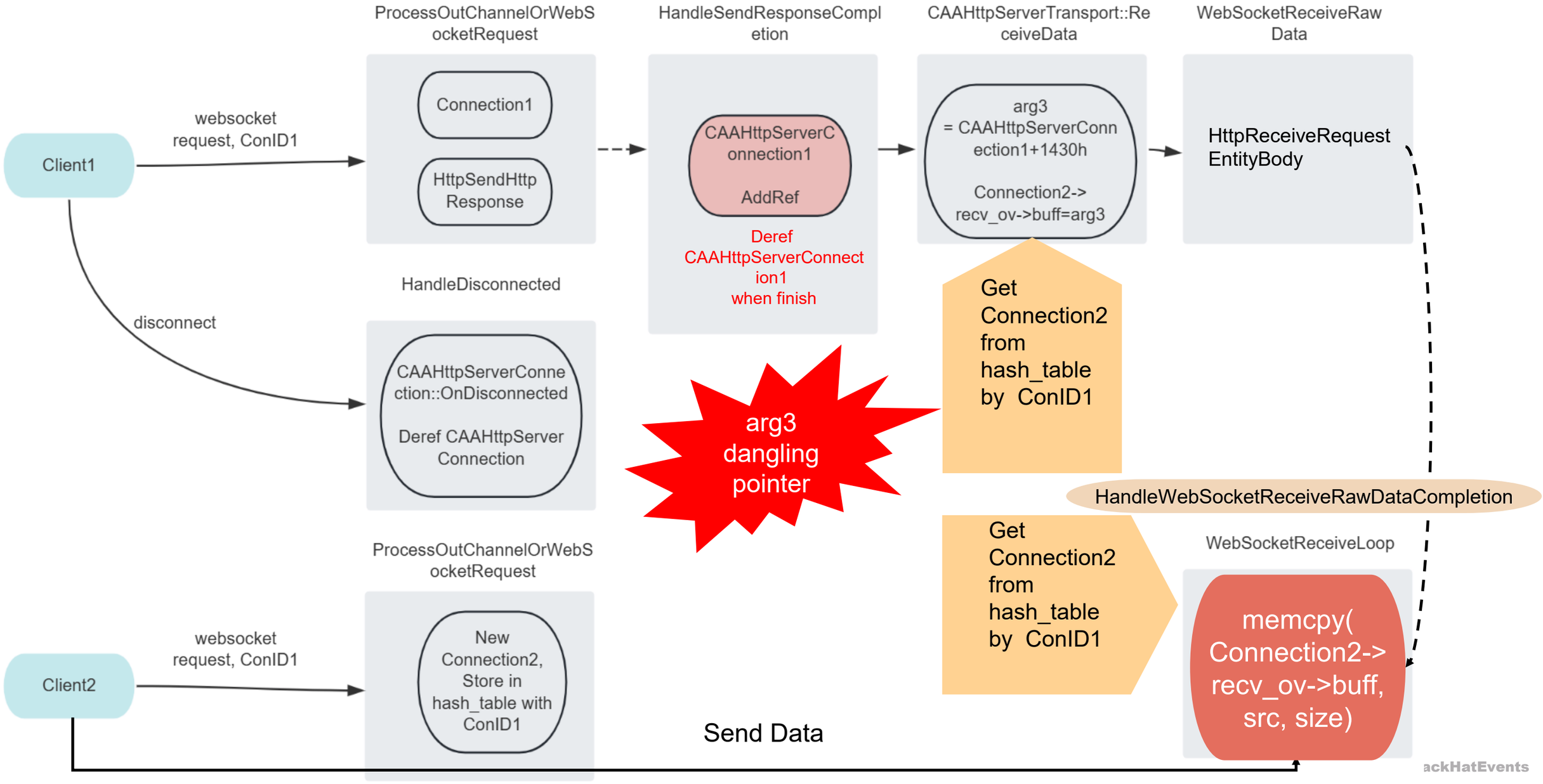

Register callback
IoCompletionCallback
Network
Websocket
non-Websocket
HttpReceiveHttpRequest
HttpReceiveRequestEntityBody
HttpWaitForDisconnect
HttpCancelHttpRequest
HttpSendHttpResponse
HttpSendResponseEntityBody
HandleWebSocketReceiveRawDataCompletion
HandleReceiveRequestCompletion
HandleReceiveRequestEntityCompletion
HandleDisconnected
HandleWebSocketSendRawDataCompletion
HandleSendResponseCompletion
HandleSendResponseEntityCompletion
new Client Connect
Client Send
Client Disconnect
Server Send Complete


HandleReceiveRequestCompletion
ProcessOutChannelOrWebSocketRequest
HandleSendResponseCompletion
CAAHttpServerTransport::ReceiveData
WebSocketReceiveRawData
Client1
websocket request, ConID1
Connection1
HttpSendHttpResponse
CAAHttpServer Connection
arg3 = CAAHttpServerConnection+1430h
Connection1-> recv_ov->buff=arg3
Clean Client1, then Client2 with ConID1?
HandleWebSocketReceiveRawDataCompletion
Send Data
WebSocketReceiveLoop
Get Connection1 from hash_table by ConID1
memcpy(Connection1-> recv_ov->buff, src, size)
Futher Authentication

Case Study -- CVE-2025-21309




ProcessOutChannelOrWebSocketRequest
HandleSendResponseCompletion
CAAHttpServerTransport::ReceiveData
Client1
websocket request, ConID1
Connection1
HttpSendHttpResponse
CAAHttpServerConnection1
AddRef
arg3 = CAAHttpServerConnection+1430h
timegap
HandleDisconnected
disconnect
CAAHttpServerConnection::OnDisconnected
Deref CAAHttpServerConnection
ProcessOutChannelOrWebSocketRequest
Client2
websocket request, ConID1
New Connection2, Store in hash_table with ConID1


ProcessOutChannelOrWebSocketRequest
HandleSendResponseCompletion
CAAHttpServerTransport::ReceiveData
WebSocketReceiveRawData
Client1
websocket request, ConID1
Connection1
HttpSendHttpResponse
CAAHttpServerConnection1
AddRef
Deref CAAHttpServerConnection1 when finish
arg3 = CAAHttpServerConnection1+1430h
Connection2->recv_ov->buff=arg3
HttpReceiveRequestEntityBody
HandleDisconnected
disconnect
CAAHttpServerConnection::OnDisconnected
Deref CAAHttpServerConnection
arg3 dangling pointer
Get Connection2 from hash_table by ConID1
HandleWebSocketReceiveRawDataCompletion
ProcessOutChannelOrWebSocketRequest
Get Connection2 from hash_table by ConID1
WebSocketReceiveLoop
Client2
websocket request, ConID1
New Connection2, Store in hash_table with ConID1
memcpy(Connection2->recv_ov->buff, src, size)
Send Data

Case Study -- CVE-2025-21309

```
0:117> r
rax=000002c6a39aabf0 rbx=000000000000000e rcx=000002c6a39aabf0
rdx=0002000000010000 rsi=000002c6ad1acf66 rdi=0000000000006000
rip=00007ffd84747d17 rsp=000000d47037f338 rbp=000000d47037f480
 r8=000000000000000e  r9=000000000000000e r10=000002c6a39aabf0
r11=0000000e00000001 r12=0000000000000000 r13=000000000000000e
r14=000002c6ee9edd60 r15=000002c6fc23bf40
iopl=0         nv up ei ng nz na pe cy
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010283
msvcrt!memcpy+0x17:
00007ffd`84747d17 4c8919          mov     qword ptr [rcx],r11 ds:000002c6`a39aabf0=????????????????
0:117> k
 # Child-SP          RetAddr           Call Site
00 000000d4`7037f338 00007ffd`84730660 msvcrt!memcpy+0x17
01 000000d4`7037f340 00007ffd`64f42c64 msvcrt!memcpy_s+0x60
02 000000d4`7037f380 00007ffd`64f43ba2 aaedge!CAAHttpServerTransport::WebSocketReceiveLoop+0xafc
03 000000d4`7037f580 00007ffd`64f455eb aaedge!CAAHttpServerTransport::HandleWebSocketReceiveRawDataCompletion+0x24e
04 000000d4`7037f610 00007ffd`8482770a aaedge!CAAHttpServerTransport::IoCompletionCallback+0x22b
05 000000d4`7037f6a0 00007ffd`85347493 KERNEL32!BasepTpIoCallback+0x5a
06 000000d4`7037f6f0 00007ffd`8534b8e8 ntdll!TppIopExecuteCallback+0x193
07 000000d4`7037f770 00007ffd`84824cb0 ntdll!TppWorkerThread+0x448
08 000000d4`7037fa60 00007ffd`853bedcb KERNEL32!BaseThreadInitThunk+0x10
09 000000d4`7037fa90 00000000`00000000 ntdll!RtlUserThreadStart+0x2b
```

Conclusion

Looking Ahead

- MSRC updated their SDL servicing bar for DoS-related vulnerabilities.(https://learn.microsoft.com/en-us/security/engineering/security-bug-bar-sample) No bounty for Resource Exhaustion 🙄

- Logic-based DoS vulnerabilities still in scope for High value assets. Includes: DHCP Server, DNS Server, epmapper (MS-RPC), Hyper-V Remote Access, IIS Web Server HTTP/HTTPs, Kerberos Authentication Service, LDAP, NFS, RDP Server, SMB, and Windows Server Update Service (WSUS).

- RCE vulnerabilities are also common in HTTP services, especially during the parsing of POST data. Try to fuzz it!

Take Aways

- Apply useful technique across the entire attack surface to uncover similar issues.
- DoS doesn't require crashes — logic flaws in request handling alone can also permanently block services
- Further reflection: the potential for DoS and even RCE may lie in the deeper, more fundamental logic of the target

Thanks!

k0shl(@KeyZ3r0) https://x.com/KeyZ3r0

VictorV(@vv474172261) https://x.com/vv474172261

Wei(@XiaoWei___) https://x.com/XiaoWei___

Zhiniang Peng(@edwardzpeng) https://x.com/edwardzpeng