USENIX SECURITY 2026 · BALTIMORE, MD, USA



Death by a Thousand Drips: Uncovering Critical Resource Leaks in the Windows Ecosystem

A resource-oriented fuzzing framework for closed-source Windows remote services

Feng Dong, Jianting Gao, Yunpeng Tian, Weifeng Yuan, Mu Zhang, Zesen Ye, Jietao Yang, Zhiniang Peng

Huazhong University of Science and Technology · The Hong Kong Polytechnic University
University of Utah · Beijing CyberKunlun Technology Co., Ltd

Artifact: doi.org/10.5281/zenodo.20623257

A remote attacker can take a service down without ever crashing it

ONE REQUEST, INSIDE THE SERVICE



× N requests — one orphaned buffer per request, and the leak never stops

SERVICE HEALTH	PROTOCOL REPLIES	PRIVATE BYTES
Responsive to every check	Normal, well-formed	Rising monotonically

> No exception. No crash. No fuzzer report.

Under the shared service host, a leak anywhere is an outage everywhere

`svchost.exe`

- **Vulnerable service** — leaks on every request
- Co-hosted service — unrelated, correct code
- Co-hosted service — unrelated, correct code
- Co-hosted service — unrelated, correct code

ONE HOST PROCESS OWNS

- One shared heap
- One address space
- One commit limit

Attribution runs the other way and breaks: monitoring tools blame generic svchost.exe, not the sub-service that allocated.

Exhaustion or fragmentation caused by any member degrades every co-hosted service

Since Windows 10 1903 many services are split into separate host processes on memory-rich clients — a performance–isolation trade-off, not a fix; consolidation remains in use on legacy, constrained and deliberately grouped configurations.

The cleanup code exists — the attacker just never lets it run

CLASS A

Release absent

Cleanup code is missing or unreachable on every path. The classic memory leak, and a well-served class.

CLASS B

Uncontrolled allocation

One oversized request asks for more than the service should ever grant. Visible in a single interaction.

CLASS C · OURS

Release bypass

Valid cleanup logic exists for standard flows, and a crafted payload reaches a path that skips it.

01	02	03	04
Allocation succeeds on the normal path	Parsing error, logic violation or failed check	Early return or control-flow deviation skips the release	Object orphaned, and no error is reported

Prior work addresses only the error-handling subset of this class — 11 of our 19 zero-days are not error-handling related.

Scope: accumulative user-mode heap exhaustion. CPU-only DoS, bandwidth saturation and kernel-pool leaks are out of scope.

Every existing leak detector fails a different prerequisite here

TOOLING FAMILY	DISQUALIFYING PREREQUISITE	STATUS
Peach, Sulley, Boofuzz	Crash oracle: no crash, no signal	✗
Valgrind, Dr. Memory	No native Windows integration; conflicts with Windows 11 hardening	✗
ASan, LSan	Need source and build-chain integration; unavailable for stripped binaries	✗
ETW	Built for human-centric audit logging; an attribution gap back to the input	✗
PageHeap, Pin	CFG conflicts, missing symbols and prohibitive overhead	✗

ONE SHARED ROOT

No source, no symbols, and hardened production binaries

AND A DEEPER GAP

Reachability detectors report the symptom, not the failing path

Treat cumulative resource consumption as a first-class signal

If one request leaks too little to see, run it enough times that the trend becomes the observable.

REQ 01

See inside closed-source Windows services

REQ 02

Amplify weak per-request effects into a slope

REQ 03

Separate a real leak from benign caching noise

REDOSPECTOR is a four-phase loop around a resource-aware oracle

PHASE 2 RUNS INSIDE THE TARGET SERVICE · CONTROLLER AND ANALYZER RUN OUTSIDE IT

PHASE 01

Seeds that pass

Reverse-engineering tools plus an LLM emit packet templates that clear validation.

PHASE 02

Heap telemetry

A DLL injected with Microsoft Detours logs every heap event with its call-site context.

PHASE 03

Amplification

Repeat each input, then watch for stair-case growth in bytes and handle counts.

PHASE 04

Proportionality

Address-level pairing and a linearity check over call stacks give a root cause.

Anomaly scores from phase 4 update the seed corpus, so the loop closes on itself

MONITOR	CONTROLLER	ANALYZER
In-process, inside the service	Drives seeds and repetitions	Offline, over a named pipe

Static artifacts plus an LLM turn a black box into a protocol-aware corpus

01 EXTRACT

From the binary

Exported functions, cross-references, string literals and switch dispatch sites in socket-facing code.

02 NORMALIZE

Into one schema

A compact, language-agnostic description per suspected parser or dispatcher, ready to put in a prompt.

03 PROMPT

GPT-4o, bounded

Two jobs: infer valid input formats, and follow function names to find call chains from an entry point.

04 MUTATE

Structure-aware

Byte-level operators plus operators that respect inferred boundaries — a length field gets values the payload contradicts.

EACH JSON TEMPLATE CARRIES

`field name` `byte offset` `length` `valid values`

Templates also carry mutation hints — boundary, rare flag, reserved. The corpus starts as these templates plus a few hand-built minimal requests, then grows with whatever previously triggered an anomaly.

THE BARRIER THIS CLEARS

`signature 0x3030 · min len 0x2A`

WHEN THE LLM IS WRONG

A misidentified field simply yields an input the service rejects early.

Hooking two ntdll routines sees both Windows heaps at once

Service code calls into the allocator

`Kernel32!HeapAlloc`

`ntdll!RtlAllocateHeap · RtlFreeHeap`
Hook point — one layer, both heaps, no symbols

NT heap | Segment heap

WHY THIS LAYER

Not VirtualAlloc or NtAllocateVirtualMemory: those are page-level, and delayed frees inside the allocator's own caches look like leaks.

STABILITY ENGINEERING

Handlers never call the instrumented heap APIs, and log into a preallocated lock-free ring buffer.

OBSERVATION TUPLE PER INTERCEPTED EVENT

$$\langle Addr, Size, TID, StackHash \rangle$$

`Addr`
Allocation base, the key that pairs a free to its allocation.

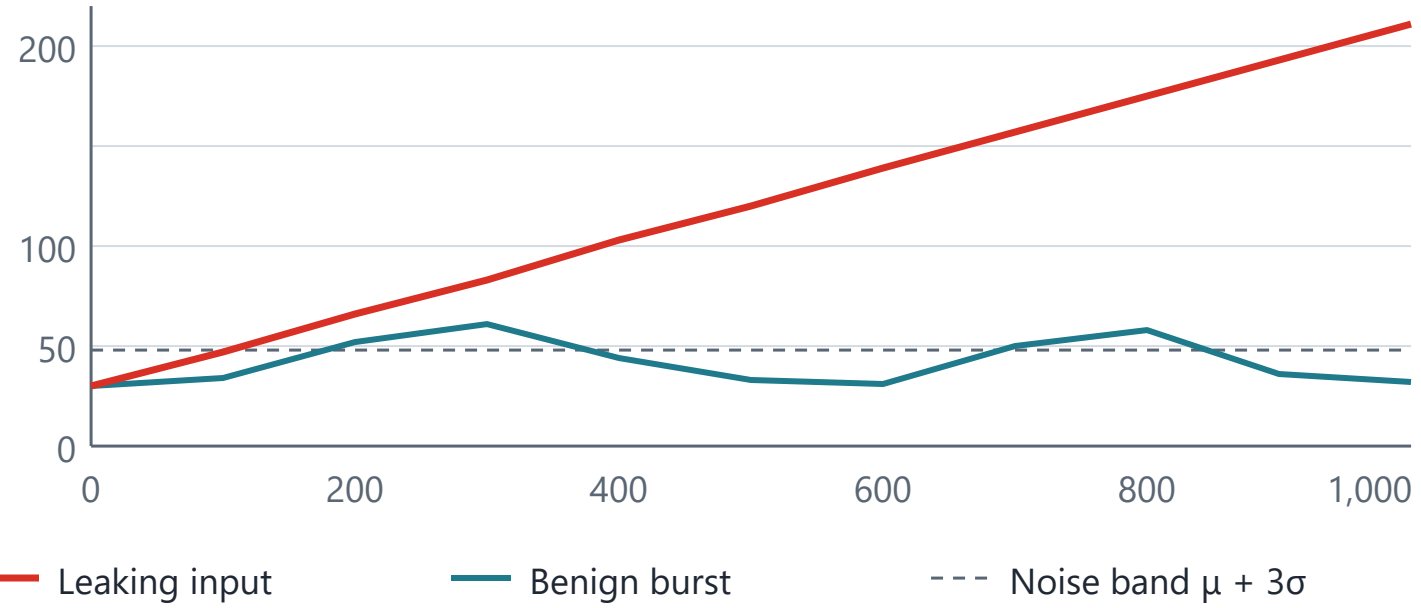
`Size`
Requested byte count, which is the leak size per request.

`TID`
Owning thread, which ties the leak to a listener or worker.

`StackHash`
64-bit hash over eight return frames, ASLR-normalized per module.

A leak is a slope that clears the noise band and never comes back down

PRIVATE BYTES IN MB OVER REPETITION COUNT · 50–200 REPEATS TO SCREEN, 1,000 TO CONFIRM



STATELESS MODE

A fresh connection per repetition

STATEFUL MODE

One long-lived session, reused

THE ORACLE FIRES ONLY WHEN

$$\beta > \mu + k \cdot \sigma$$

01 SLOPE

The slope clears the noise band, with k = 3 by default.

02 SHAPE

Usage rises per packet and never falls after the reset.

03 CALL SITE

Orphaned allocations pile up at one consistent site.

If every request leaks the same object, the count must scale with n

PHASE 01

Pair every free to its allocation

For every allocation tuple, look for a free record on the same address. What remains unmatched is orphaned.

PHASE 02

Aggregate by call-stack fingerprint

Group orphans by StackHash and count per stack. One fingerprint is one place in the binary where objects go missing.

PHASE 03

Linear proportionality check

The leak index should land near a small integer C, inside a tolerance band ε = 0.1. Filter one drops C < 1 as sub-linear noise.

$$L_\sigma = \frac{|\text{Orphaned}_\sigma|}{n}, \qquad \big||\text{Orphaned}_\sigma| - C \cdot n\big| \le \varepsilon \cdot n$$

Benign packets go first: initialization stacks below that baseline are dropped

Clustering and ranking then reduce triage effort — they do not decide truth

12,500 lines of C and C++, about 20% overhead, 48 hours per target

CODE SIZE

12.5k

lines of native 64-bit C/C++

RUNTIME COST

~20%

on average, analysis offline

CAMPAIGN BUDGET

48 h

wall clock, per target service

THREE PIECES

- In-process monitoring DLL
- Detours-based launcher
- Central controller

EVALUATION CONFIGURATION

Windows 11 24H2 and Windows Server Preview, on VMware with 4 vCPU and 8 GB, snapshot-restored between campaigns. Threshold k = 3, amplification n = 1,000.

A new target is one YAML file — service name, binary path, arguments, endpoints, authentication — plus a harness; templates ship for CDP and MSMQ. Beyond n = 1,000 the returns diminished.

19 previously unknown release-bypass vulnerabilities in core Windows services

ZERO-DAYS

19

previously unknown

ASSIGNED CVES

12

reported Dec 2024

ARCHITECTURAL FIX

7

stack redesigned

SERVICE	VULNERABLE MODULES	BUGS	DISCLOSURE
CDP Service	`cdp.dll, cdpsvc.dll`	7	Stack redesigned
Message Queuing	`mqqm.dll, mqise.dll`	6	CVEs assigned
Storage Management	`concrete.dll`	4	CVEs assigned
Remote Desktop Gateway	`aaedge.dll`	1	CVE assigned
Deployment Services	`wdstftp.dll`	1	CVE assigned

TARGET SELECTION

Default-enabled or commonly active services, underexplored in the literature — SMB deliberately excluded — across deployment, storage, cloud and HMI stacks

BEYOND PRIOR WORK

11 of the 19 are not error-handling related at all.

VENDOR RESPONSE

Fixes shipped in staged updates from January to June 2025.

Six to nine minutes from first request to loss of availability

SERVICE	HOSTING PROCESS	REQUESTS	TIME	FAILURE MODE
CDP Service	svchost.exe	4.1M	6 min	Outage or crash
Remote Desktop Gateway	svchost.exe	204M	9 min	Outage or crash
Deployment Services	svchost.exe	8.3M	7 min	Outage or crash
Storage Management	svchost.exe	5.7M	9 min	Outage or crash
Message Queuing	mqsvc.exe	14M	8 min	Outage or crash

WHAT OUTAGE MEANS

Persistent unresponsiveness, repeated failures or refusal of new connections.

RECOVERY

The host process need not terminate — recovery may need a service restart.

READ WITH CARE

Thresholds depend on hardware, service configuration and whether the target shares a svchost.exe.

Boofuzz found zero — the oracle is the difference

	REDOSPECTOR		BOOFUZZ, SAME 48 HOURS	
SERVICE	CRASHES	UNIQUE BUGS	CRASHES	UNIQUE BUGS
CDP Service	1,423	7	0	0
Remote Desktop Gateway	672	1	0	0
Deployment Services	621	1	0	0
Storage Management	855	4	0	0
Message Queuing	1,389	6	0	0

WHAT WAS HELD CONSTANT

Same seed corpora, same mutation strategy, same target services, and length fields aligned through Boofuzz's native Size element.

THE ONLY DIFFERENCE

No in-process heap monitor. The services stayed responsive while leaking.

Amplification alone is not enough — proportionality is what makes it usable

ONE-HOUR RUN ON CDPSVC · GROUND TRUTH BY MANUAL VERIFICATION

ORACLE CONFIGURATION	ALERTS	FALSE POS.	PRECISION
Naive threshold	1,402	1,401	< 0.1%
Amplification only	145	121	16.5%
Amplification + LPC	**21**	**1**	**95.2%**

WHAT THAT BUYS

1,402 → 21

alerts per hour of runtime

20 of the 21 were real leaks

WHY THE MIDDLE ROW FAILS

Benign caching grows monotonically but not linearly. LPC demands a strict linear relation at one call site, and the benign-packet baseline removes the initialization residue.

THE OTHER ABLATION AXIS

Without protocol-aware seeds, format-agnostic mutations fail the signature, minimum-length and length-consistency checks — they never reach the core logic to be judged at all.

CVE-2025-33068 — 0x820 bytes per request, forever

MSStrgSvc.exe → concrete.dll → HTTPListener::ValidateClientCertificate

DECOMPILED, SIMPLIFIED

```
buf = realloc(buf, 0x820);
st = HttpReceiveClientCertificate(req, ...);
if (st == 0x490) // element not found
return st; // buf is never released
```

WHAT HAPPENS NEXT

A status code meaning element not found steers execution into an early return. The rate is constant, so the process exhausts its address space and crashes.

PER REQUEST, NEVER RELEASED

0x820

CALL STACK INSIDE CONCRETE.DLL

```
ValidateClientCertificate+0x8f
HTTPListener::Run+0x103
ThreadProc+0x109
```

Reachable when client certs are required.

CDP pre-auth DoS — bypassing the CVE-2025-21207 fix by changing error path

CDPSvc → cdp.dll → MessageAccumulator::AccumulateMessageData

DECOMPILED, SIMPLIFIED

```
if (msgLen < 0x2A) { // base header size
ex = MakeAnomalyException(...);
// no TcpTransport::Disconnect here
cdp::CdpThrow(ex);
}
```

WHY THE EARLIER PATCH DID NOT HOLD

The same class re-emerges on a sibling error path — reachable pre-auth.

MICROSOFT'S RESPONSE

Redesign the communication stack rather than patch instances one by one.

THE BUG IS AN ABSENCE

The malformed packet is correctly identified. The handling of it is what leaks the session.

FIVE FRAMES, ONE WORKER POOL

```
AccumulateMessageData
GetMessagesFromData+0x107
OnTransportReceivedDataInternal
ProcessWorkitems+0x1f1
ThreadPoolProc+0x48
```

The honest scope: user-mode heap, offline analysis, imperfect inference

01 RESOURCE SCOPE

User space, per request

Purely kernel-side leaks and CPU-only DoS are out of view. Extending to more APIs and counters is straightforward future work.

02 HEAP AMBIGUITY

Pooling can look like leaking

Allocator reuse weakens address pairing, and one site may serve several object types. Highly async services make attribution harder still.

03 INFERENCE QUALITY

Wrong fields waste budget

Output quality tracks the completeness of the extracted binary features. Feeding runtime results back to refine templates is the obvious next step.

04 AMPLIFICATION COST

Sensitivity costs throughput

Secondary crashes and timeouts complicate triage. Two-stage screening then confirmation balances it, but the optimum stays target-dependent.

Orthogonal to coverage-guided fuzzing, but amplification changes a testcase's cost model — decouple exploration from confirmation. Run in isolated VMs; endpoint security products may interfere with DLL injection or heap instrumentation.

Cumulative resource growth is a fuzzing signal you are probably not collecting

THE RECIPE, IN THREE MOVES

- Amplify the weak per-request effect
- Pair allocations to frees by address
- Demand linear proportionality per stack

Run cheap coverage-driven fuzzing most of the time, then subject promising inputs to the amplified resource oracle periodically.

WHAT IT BOUGHT HERE

19 zero-days and 12 CVEs in default-enabled services.

Zero for a crash-centric baseline under the same 48-hour budget, and a vendor stack redesign.

ARTIFACT

`https://doi.org/10.5281/zenodo.20623257`

Hook DLL and injector, five harnesses, Boofuzz comparison scripts, seed templates and LLM prompts.
Withheld on purpose: pre-built binaries and exploit payloads — patch deployment lags disclosure.

A thousand drips is a measurement problem before it is a memory problem.

Thank you — questions welcome

Happy to go deeper on the oracle, the LPC threshold, or the disclosure timeline.

ARTIFACT

`https://doi.org/10.5281/zenodo.20623257`

ACKNOWLEDGMENT

Supported by the National Natural Science Foundation of China, grant 62302181. Mu Zhang was supported in part by NSF OAC-2530911 and by Battelle Energy Alliance, LLC under contract 313367.