THALIUM

Fuzzing RDPEGFX with *what the fuzz*

Colas Le Guernic, Jérémy Rubert, and Tomme of Normandy

October 15th, 2022

/HEXACON/

Who are we?

Colas Le Guernic

Jérémy Rubert

Tomme of Normandy

THALIUM team members

(part of THALES)

Building a future we can all trust

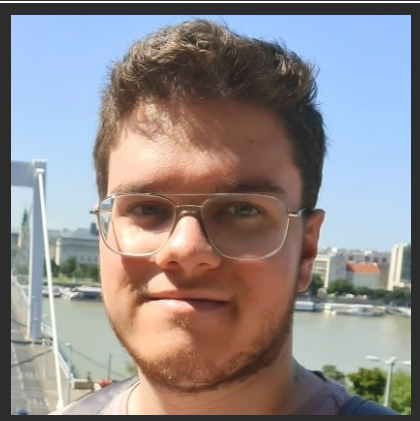
| reverse | vulnerability research | red team | **hiring** |

Valentino's 2021 internship

Coran Le Guerroué

Jérémy Rubert

Tomme



Valentino Ricotta, 2021 intern

and soon a full time Thalium member

Valentino's 2021 internship

- Fuzz Remote Desktop Protocol **clients**
  - WinAFL + network-level approach for the harness
- build over a BH Europe 2019 talk by Park et al.:

> “ Fuzzing and Exploiting Virtual Channels in Microsoft RDP for Fun and Profit ”

- Results:
  - **4 CVEs** (2 Microsoft, 2 FreeRDP)
  - thalium.re blog posts and SSTIC 2022 talk

Valentino's 2021 internship

- Several limitations:
  - some channels could not be fuzzed
  - unknown protocol state (hard to reproduce)
  - availability (restart on malformed message)

- *@0vercl0k* had just released *what the fuzz* (wtf):
  - a promising snapshot fuzzer
- we had a few weeks off

Let's try to fuzz MS RDP client with *wtf* !

Outline

- RDP and the EGFX channel
- *what the fuzz* snapshot fuzzer
- Our fuzzing campaign
  - basic harness / snapshot
  - modified harness / coverage
  - crash / analysis / CVE-2022-30221

Detailed blog post on thalium.re

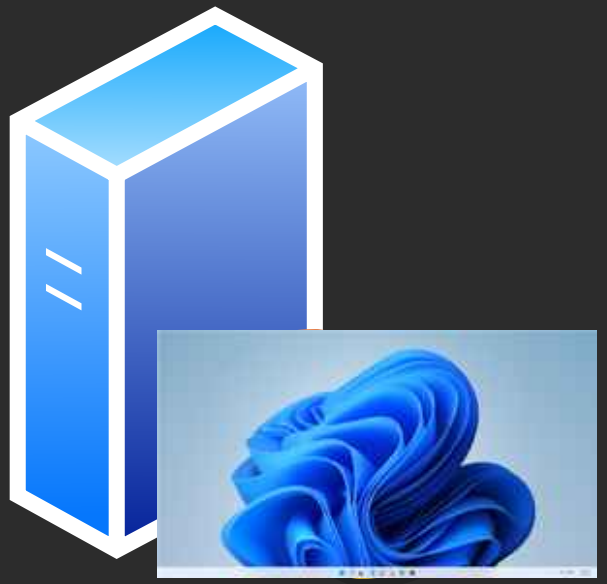
Microsoft Remote Desktop Protocol

mouse / keyboard

Remote Desktop Connection

audio / graphics

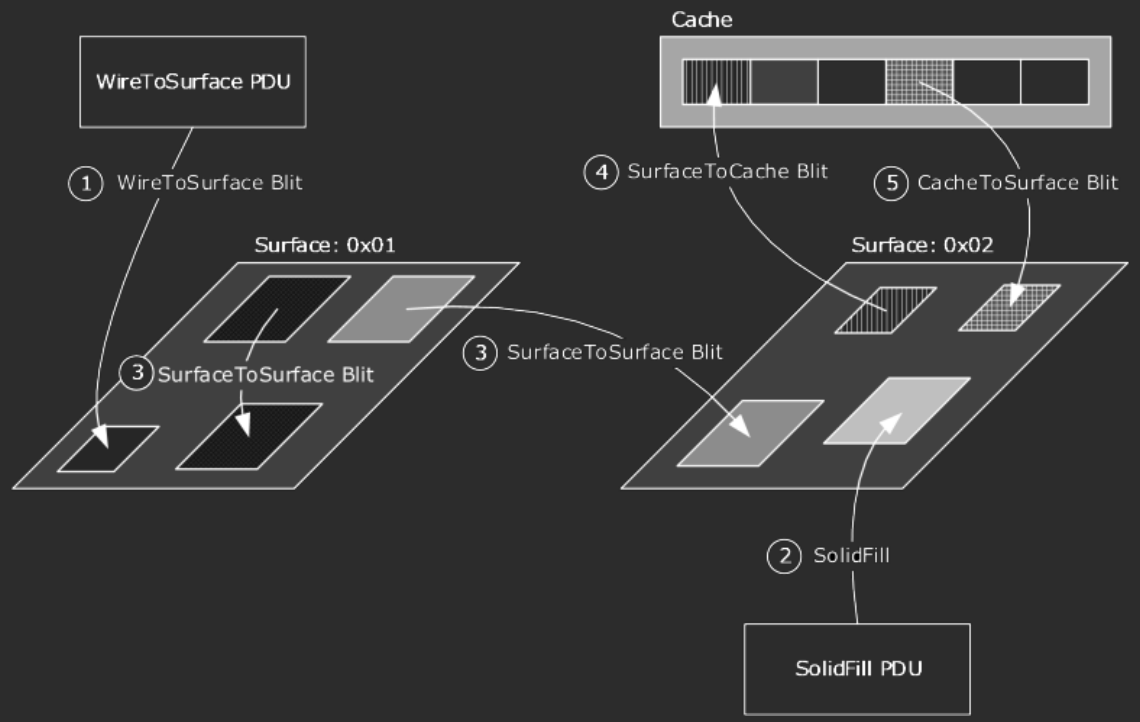
clipboard/printer/smartcard/...



Documented through Microsoft Open Spec. program

RDPEGFX: Graphics Pipeline Extension

Efficiently encode and transmit graphics display data from the server to the client

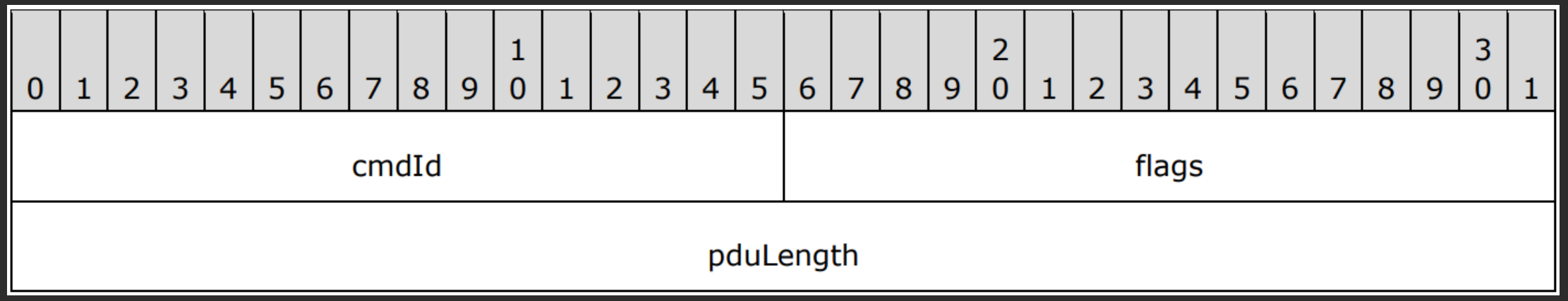


- ~asynchronous protocol
- 1 of many virt. channels
- ~20 commands (PDU)
  - surface/cache/blit/...
- multiple PDUs per msg

source:
docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpegfx/

RDPEGFX: PDUs

- Common Header:



- Different body for each `cmdId`

WhatTheFuzz

Snapshot fuzzer by *@0vercl0k* available at

https://github.com/0vercl0k/wtf

snapshot --> harness --> fuzz!!!

wtf

- snapshot
  - kernel debug, break on data processing, dump
  - no more disk / IO
- harness
  - breakpoint based
  - patch memory/registers to inject fuzz samples
- fuzz
  - **run**: emulator or hypervisor backend
  - **restore** cpu / dirty pages
  - **repeat**

Backends

Bochscpu

- full emulation
- collect all `rip` executed
- slow but powerful
- can log tenet traces

KVM/Hyper-V

- virtualization
- one-time breakpoints for coverage
- fast

`bp` on a memory dump

KVM/Hyper-V need breakpoints to register coverage

- **Virtual Addresses** of basic blocks
- Hardware Virtual Address Translation (VAT)
  `CR3` ➡️ ... ➡️ Page Table Entry (PTE) ➡️
  `PTE.Valid == 1` ➡️ **Physical Addr**

--> done by *wtf*

`bp` on a memory dump

- `PTE.Valid == 0` ➡️ Many cases
  - 😱 not implemented in *wtf*
  - needed to set BP and can not handle cases accessing disk
  - Implemented in *wtf* PR#136

Or read documentation and use lockmem

But who reads docs anyway?

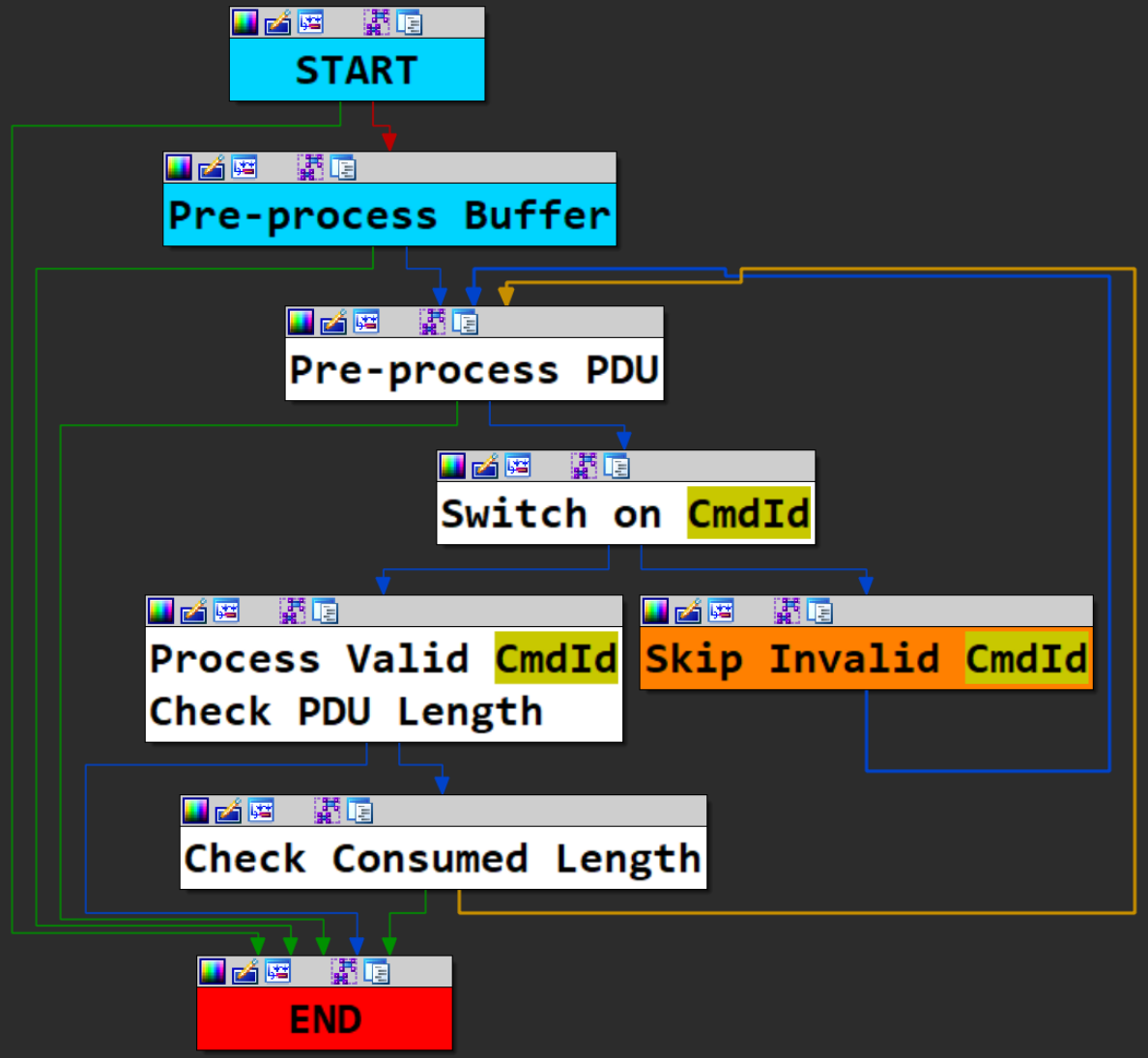
Fuzzing campaigns

- First campaign
- Tweaking the Harness
- Tweaking the Coverage
- Crashes

Harnessing RDPEGFX

```
mstscax.dll!
RdpGfxProtocolClientDecoder::Decode
```

Harnessing RDPEGFX

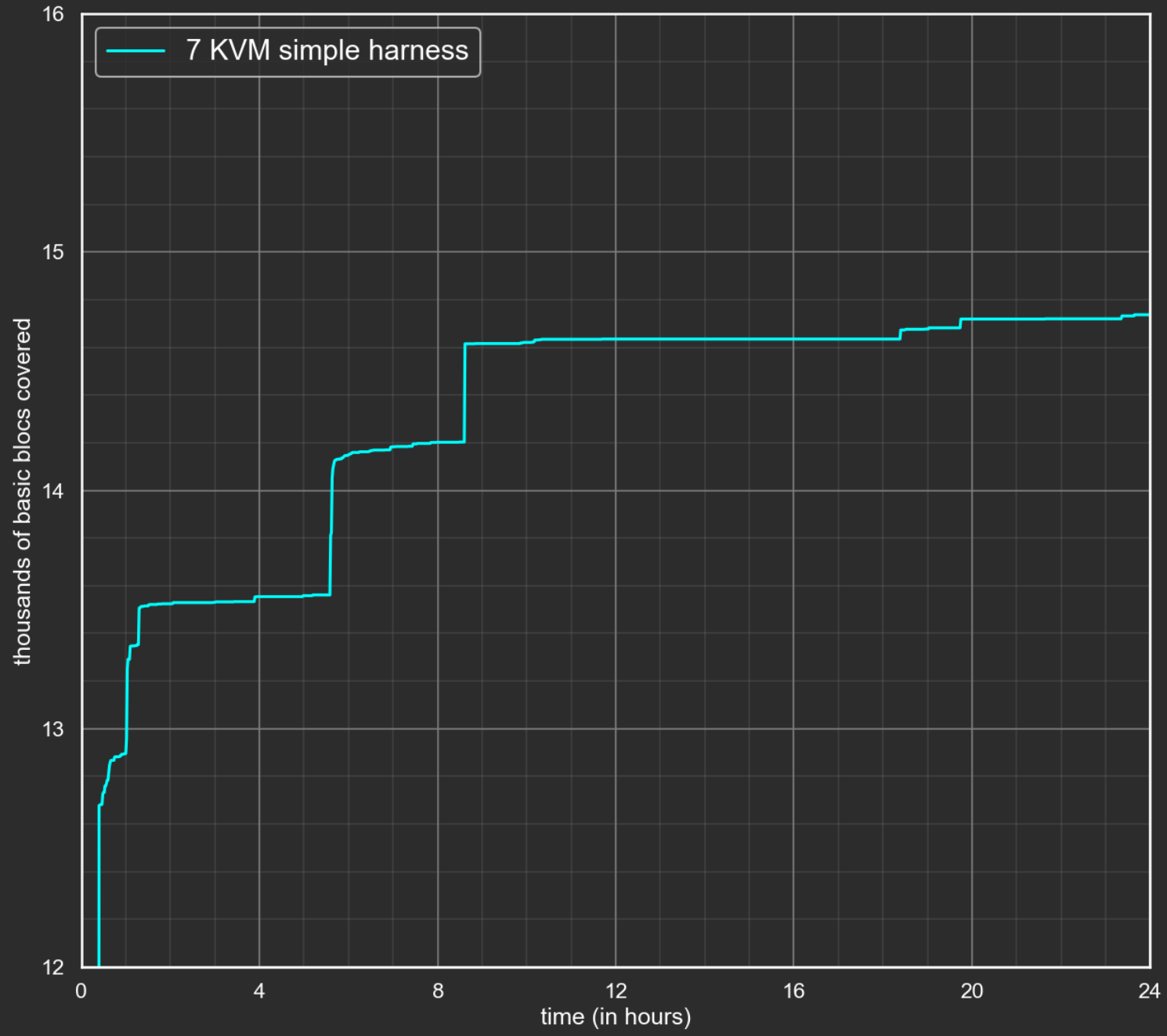


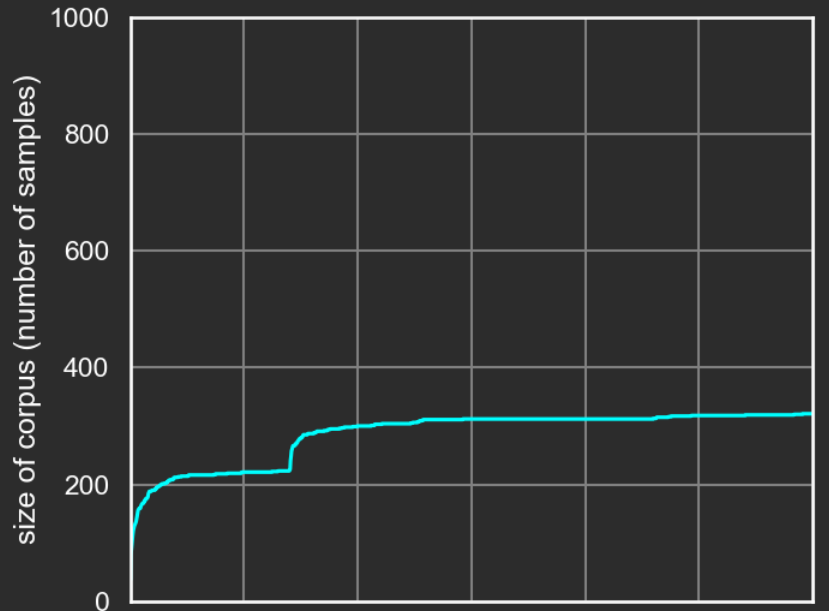
- 1 buffer ➡️ many PDUs
- 1 PDU ➡️ 1 of 20 cmds
- dispatch to 20 handlers

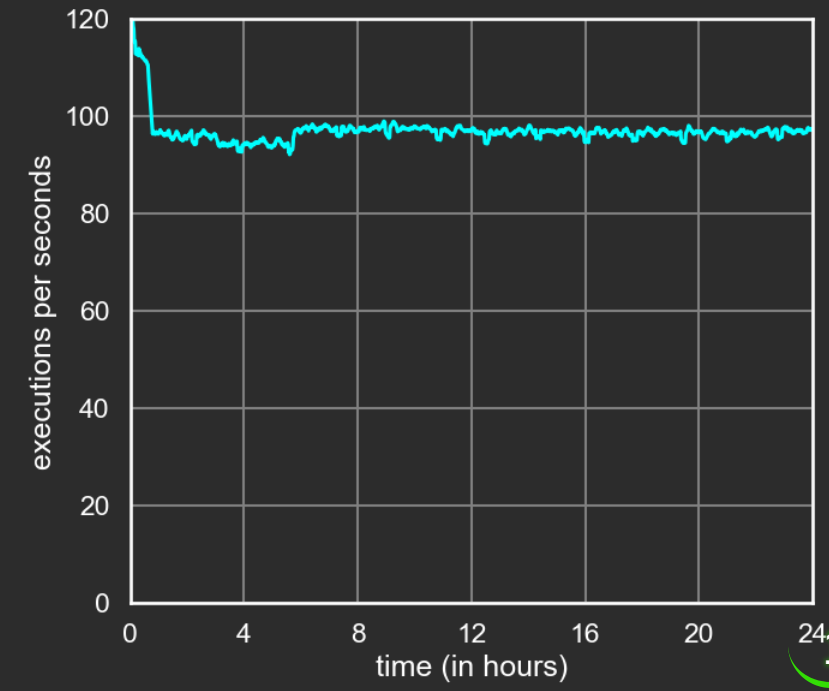
- Hook before PDU loop
  - capture corpus
  - dump snapshot
  - inject sample

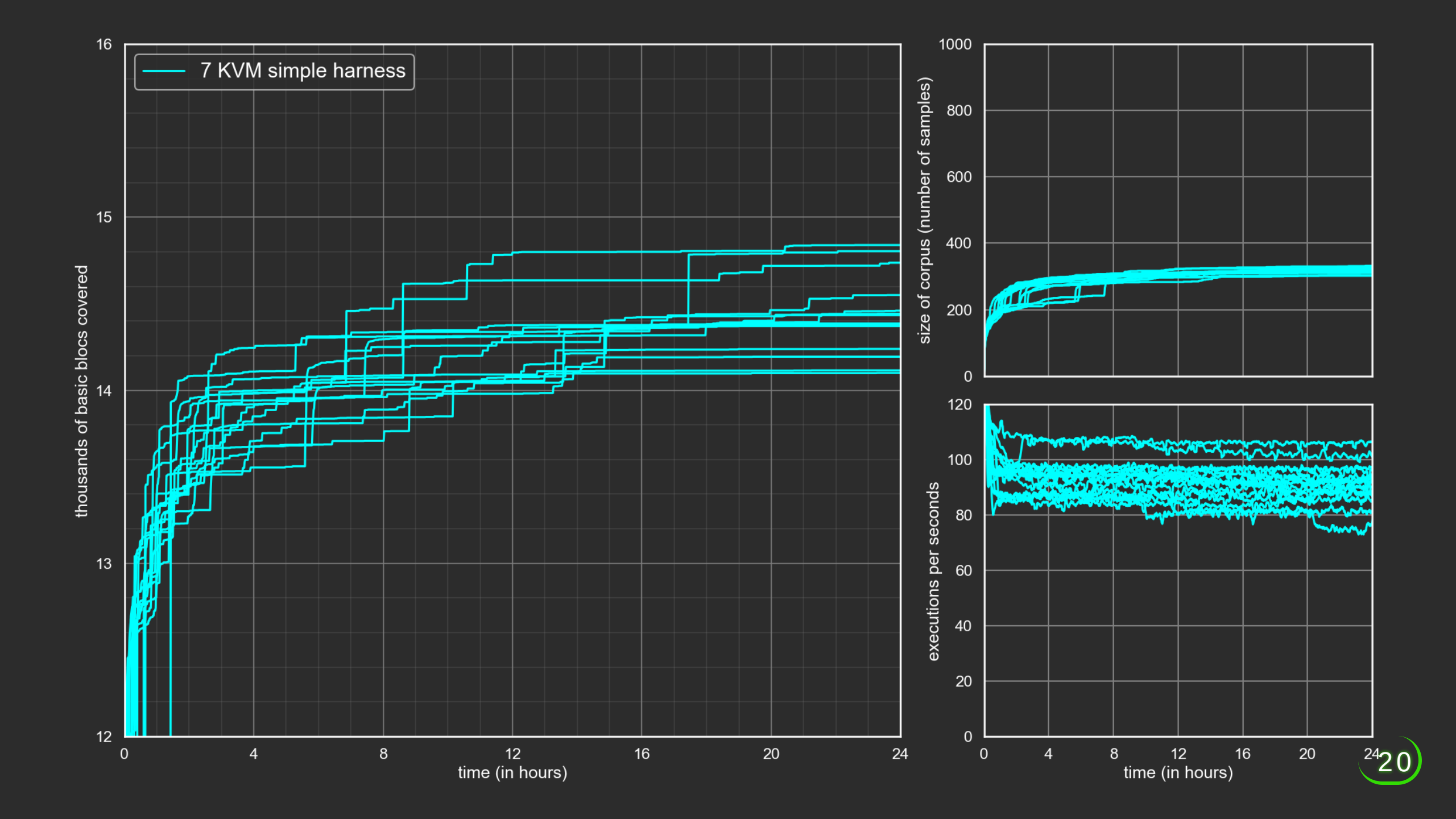
First fuzzing campaign

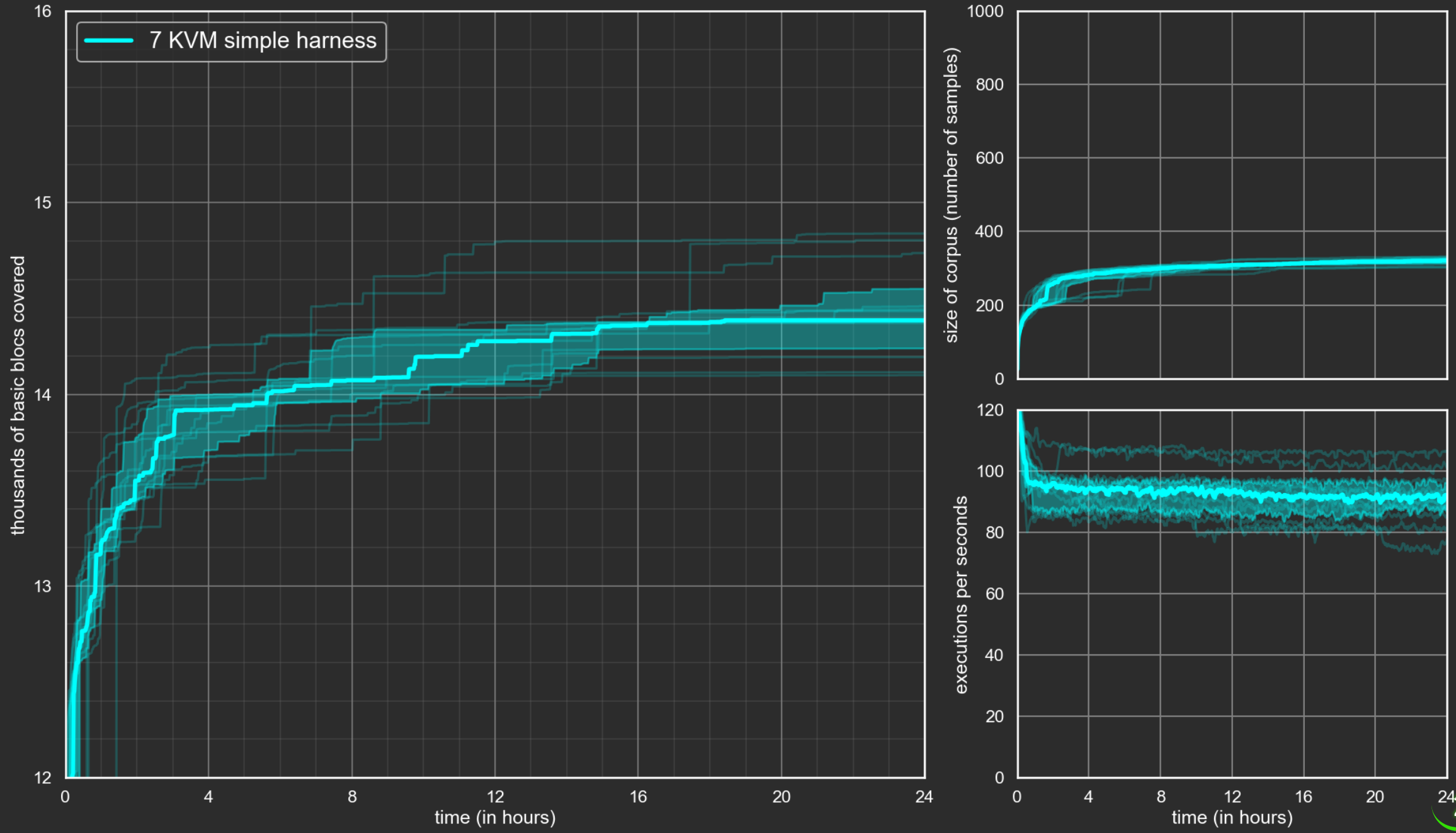
- Capture 989 messages from live RDP sessions
  - 90% smaller than 22KB
- Generate dump
  - Generate correct dump, then a better one...
- Deactivate ETW
- Hook I/Os
  - performance counters, logs...
- Make sure crash detection works


7 KVM simple harness
thousands of basic blocs covered
16
15
14
13
12
0
4
8
12
16
20
24
time (in hours)


size of corpus (number of samples)
1000
800
600
400
200
0


executions per seconds
120
100
80
60
40
20
0
0
4
8
12
16
20
24
time (in hours)


7 KVM simple harness
thousands of basic blocs covered
time (in hours)
size of corpus (number of samples)
executions per seconds
time (in hours)


7 KVM simple harness
thousands of basic blocs covered
16
15
14
13
12
0
4
8
12
16
20
24
time (in hours)
size of corpus (number of samples)
1000
800
600
400
200
0
executions per seconds
120
100
80
60
40
20
0
0
4
8
12
16
20
24
time (in hours)

Improving campaigns and Evaluation

Main tweaks

- better harness
  - on the fly fixes
- richer coverage
  - context sensitive edge
  - dirty

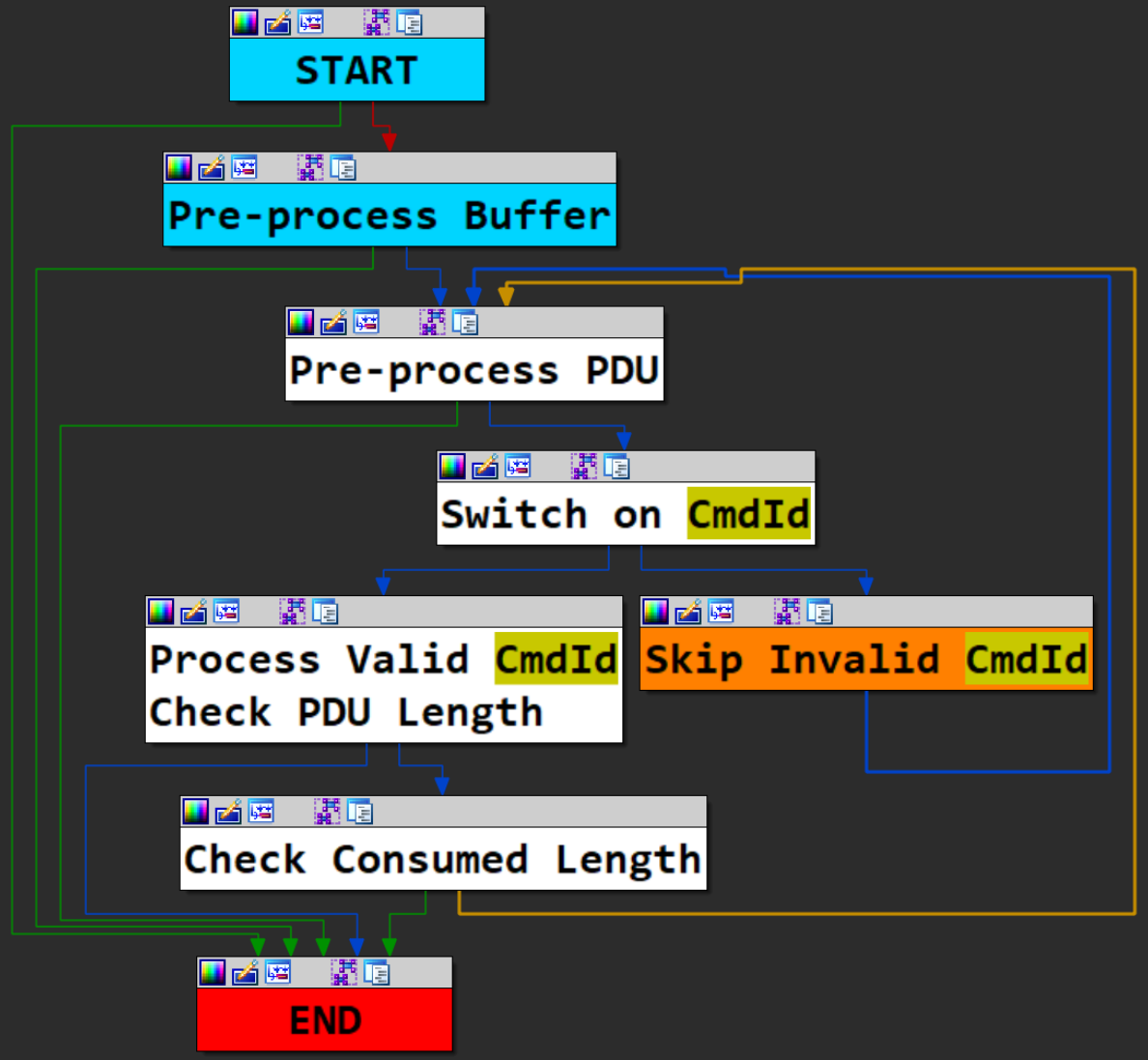
Other Tweaks

- *premature exit*
- *exotic coverages*
  - `imul` *overflow*
  - *timing*
- *corpus tweaks*
- *...*

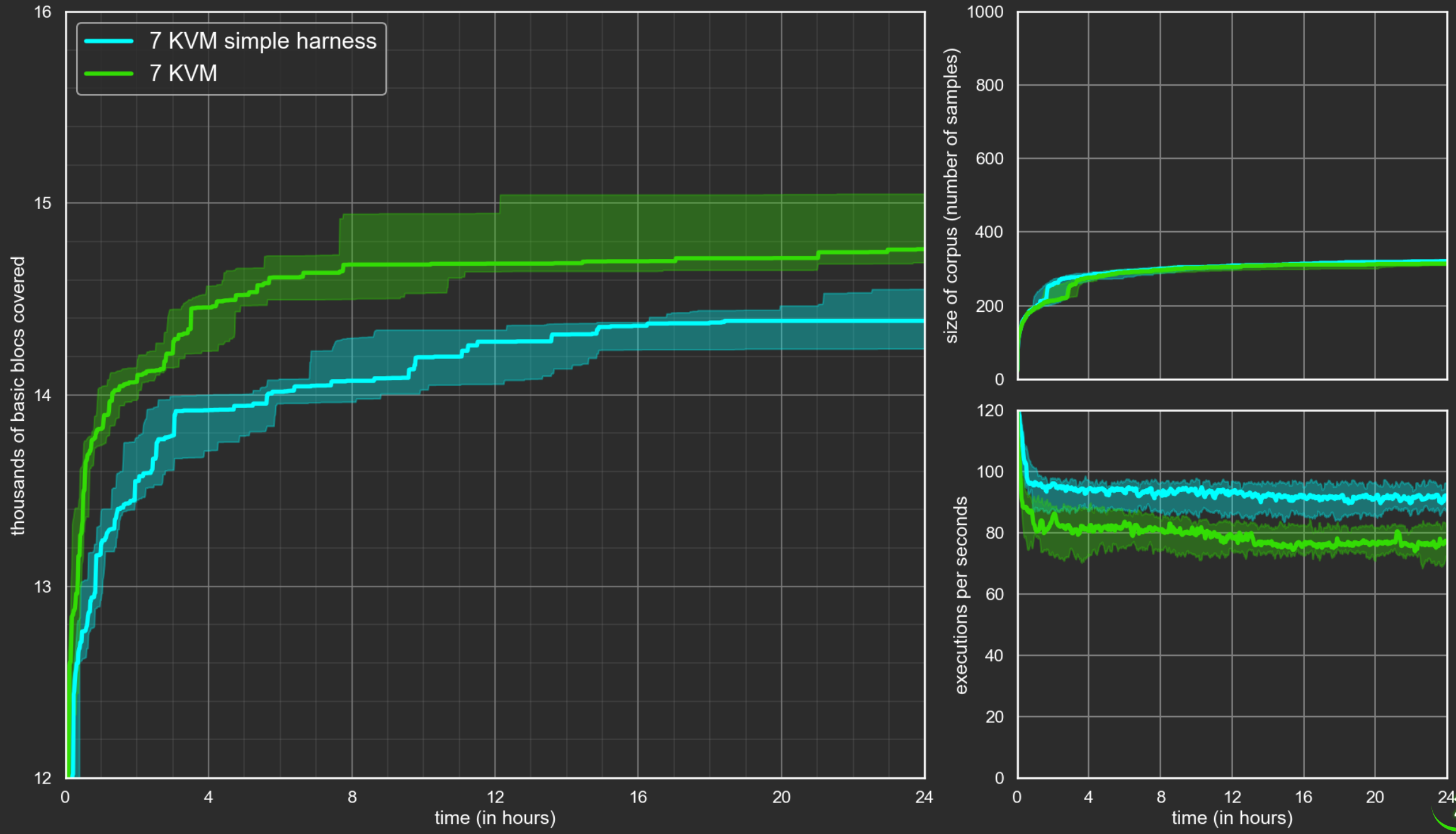
More details in our blog: thalium.re

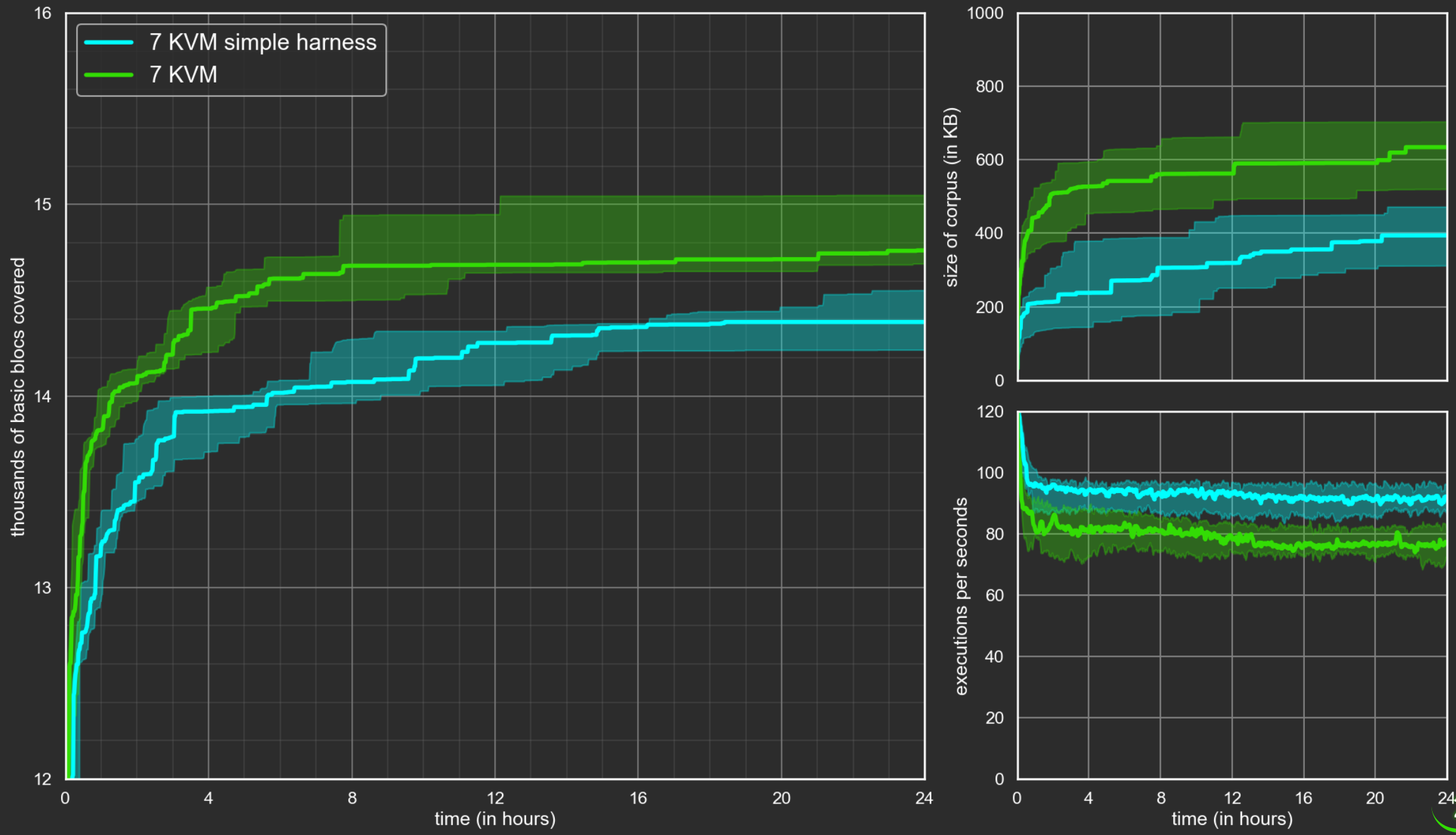
Tweaking the Harness

Tweaking the Harness



- On the fly modification of the sample
  - supply valid `CmdId`
  - supply valid `length`
- Patch target
  - Skip `length` check


7 KVM simple harness
7 KVM
thousands of basic blocs covered
16
15
14
13
12
0
4
8
12
16
20
24
time (in hours)
size of corpus (number of samples)
1000
800
600
400
200
0
executions per seconds
120
100
80
60
40
20
0
time (in hours)


7 KVM simple harness
7 KVM
thousands of basic blocs covered
time (in hours)
size of corpus (in KB)
executions per seconds
time (in hours)

Tweaking the Coverage

Edge Coverage

- With Bochscpu, as simple as registering a callback:

```
void bx_instr_cnear_branch_taken(unsigned cpu, bx_address src_rip,
                                 bx_address dst_rip)
```

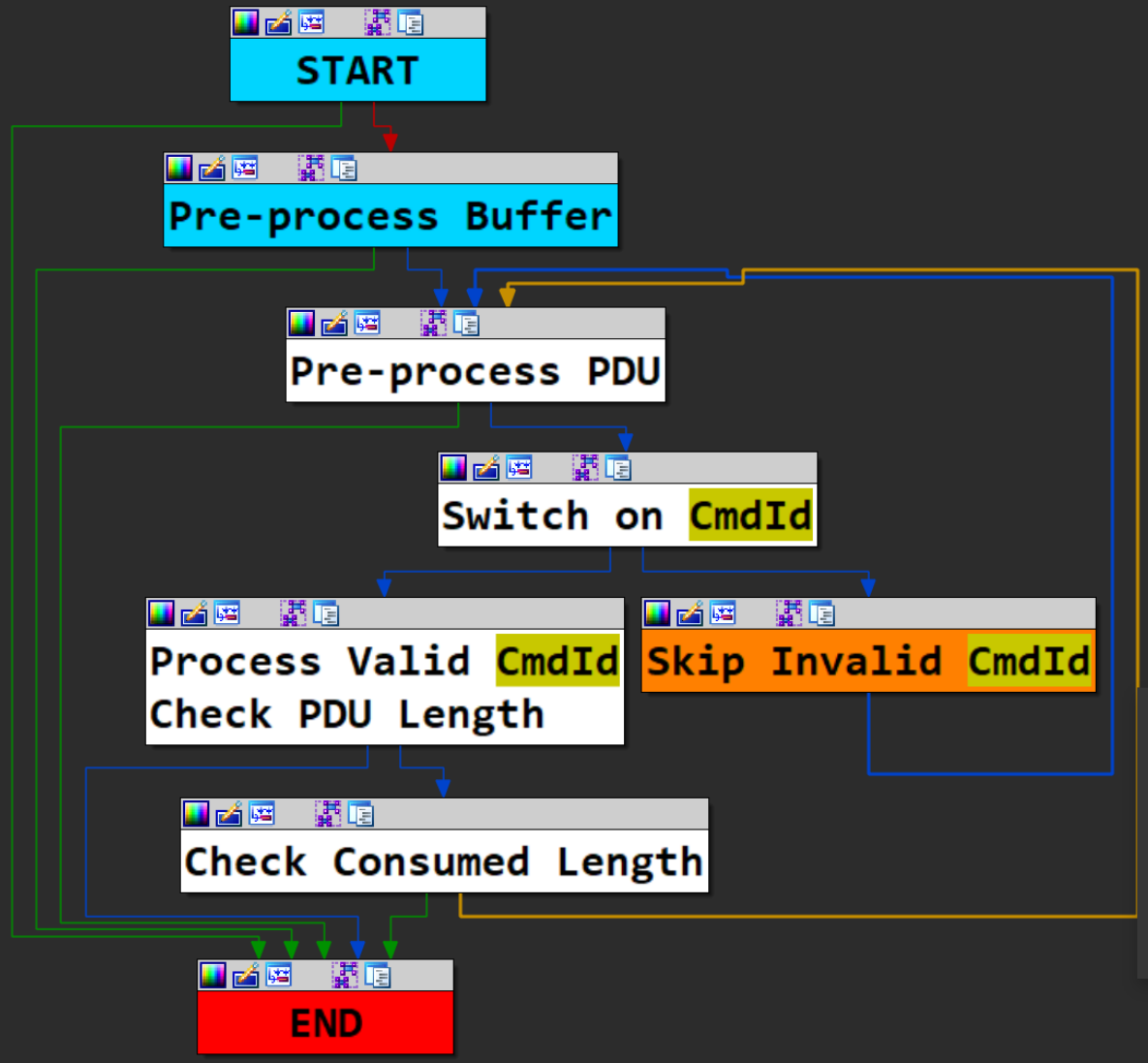
> *The callback is called each time, when currently executed instruction is a conditional near branch and it is taken.*

- calling:

```
RecordCoverage(hash(src_rip) ^ dst_rip);
```

- similarly for `bx_instr_cnear_branch_not_taken` and `bx_instr_ucnear_branch`, implemented in PR#137

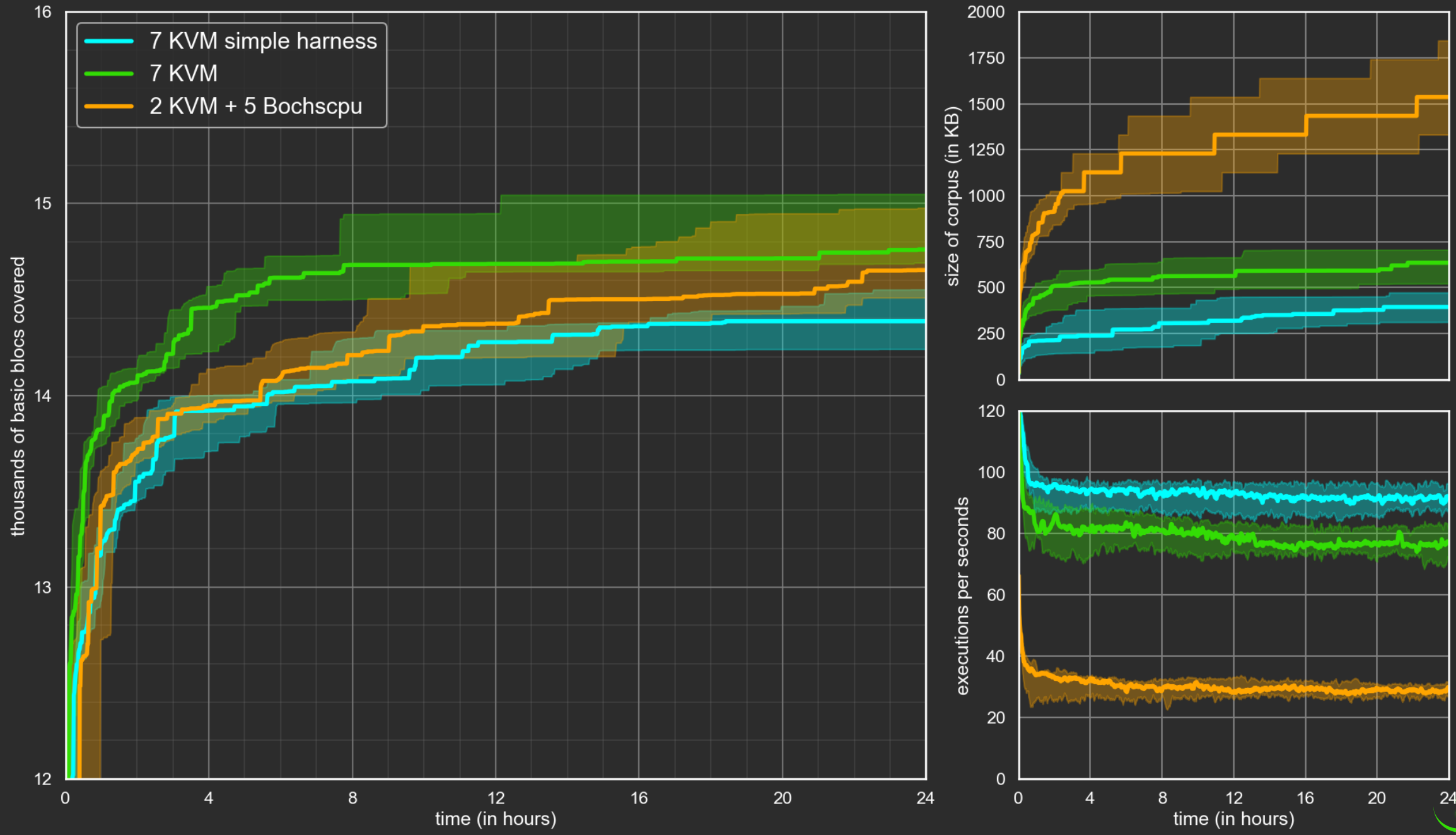
Context Sensitive Coverage



- new public attribute:
  - `Backend->state`

- `Backend->state = CmdId`

```
void RecordCoverage(uint64_t x)
{
  Cov.emplace(hash(x) ^ state)
}
```


7 KVM simple harness
7 KVM
2 KVM + 5 Bochscpu
thousands of basic blocs covered
time (in hours)
size of corpus (in KB)
executions per seconds
time (in hours)

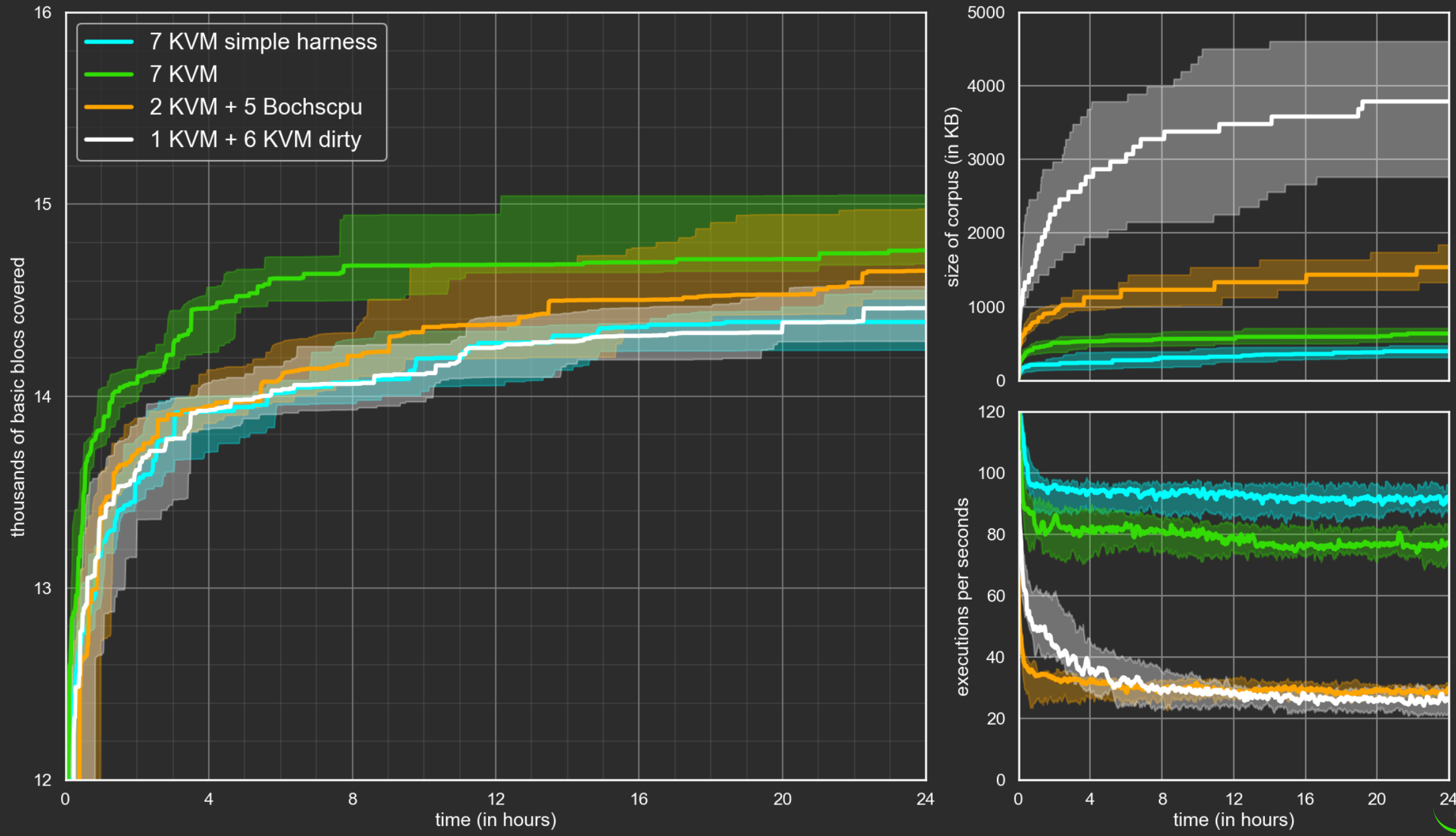
What about other Backends?

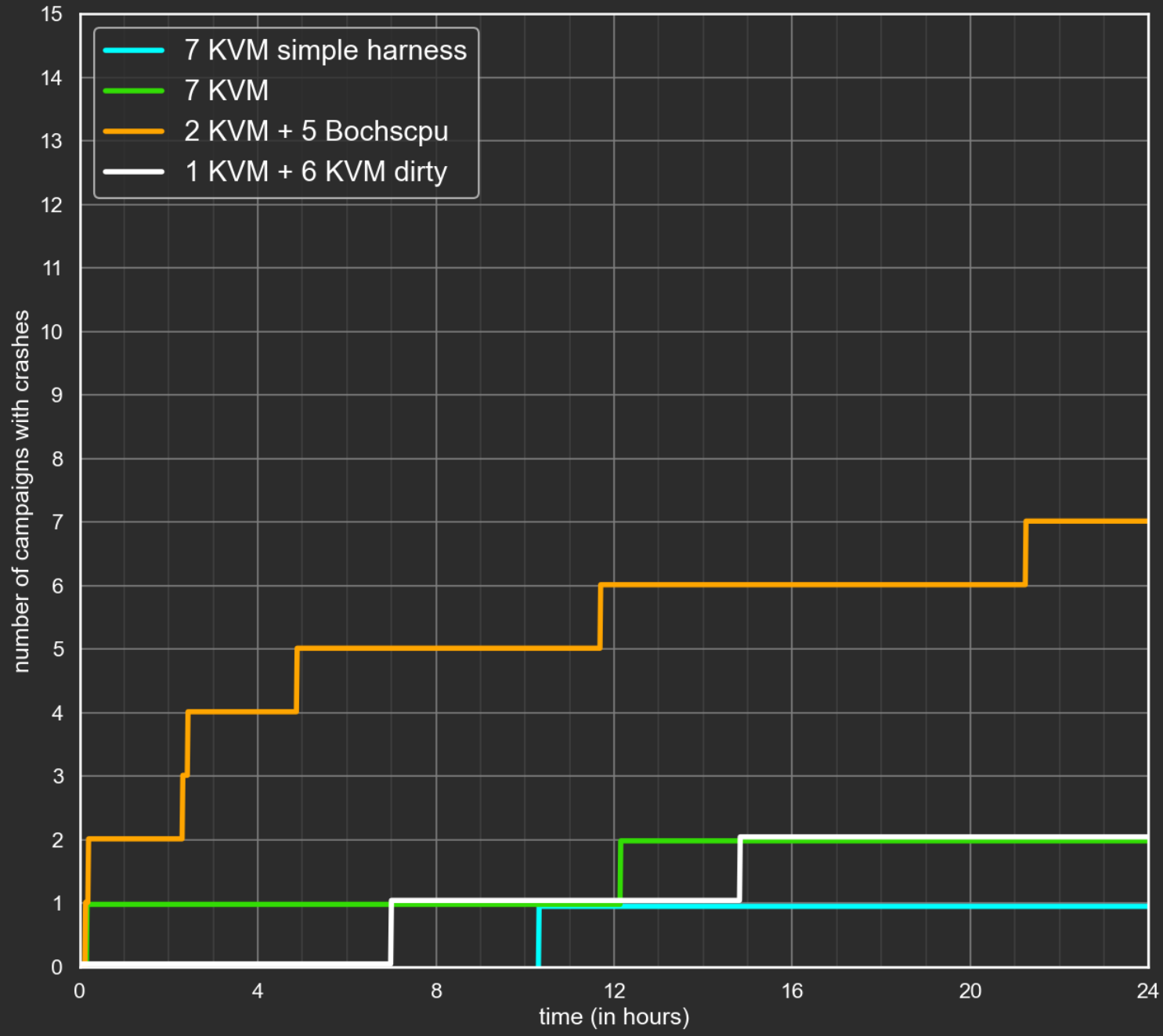
- Basic blocks coverage with temporary breakpoints
- **Not *easily* extensible** to edge coverage

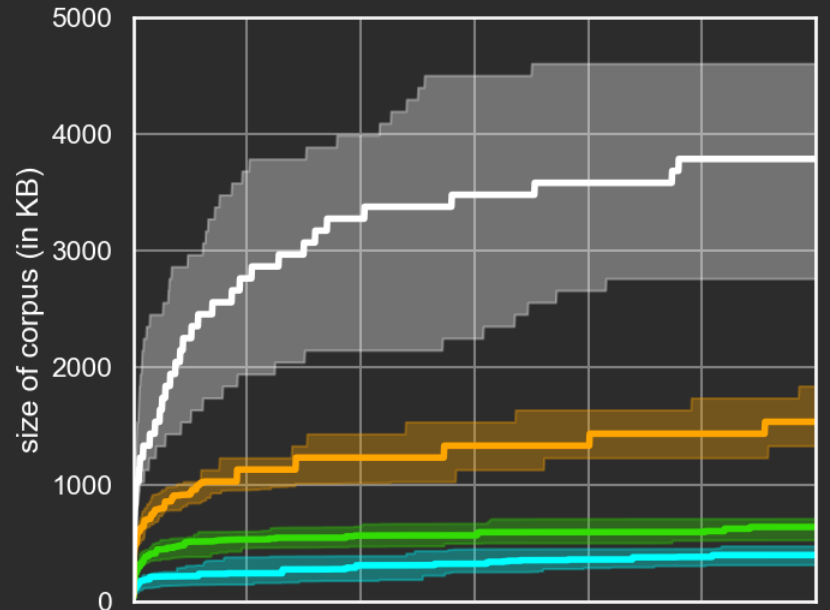
- What other behavioral information can we use?
  - *dirty pages* are already available for **free**

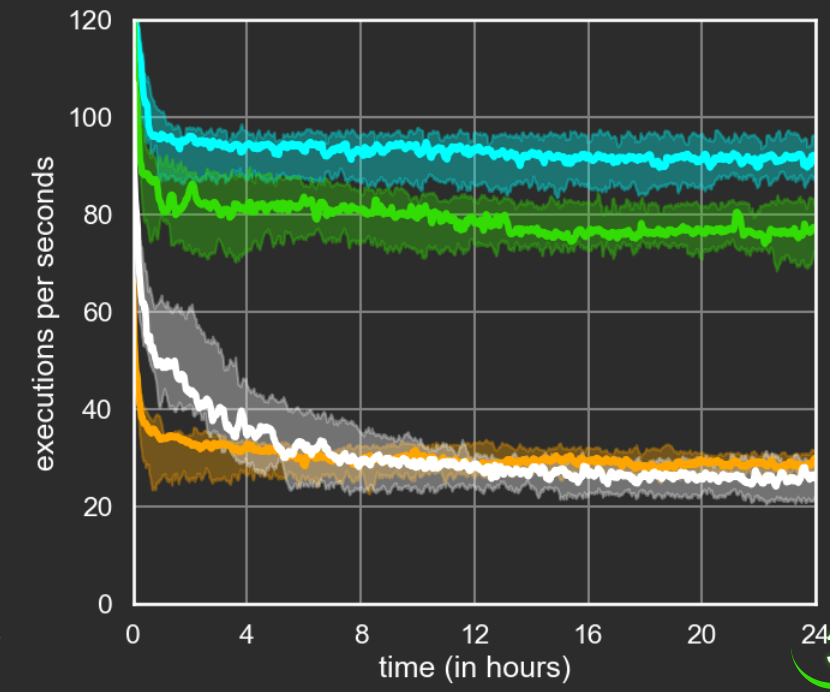
```
for (const auto &DirtyGpa : DirtyGpas_) {
  Cov.emplace(DirtyGpa);
}
```

  - side effect: favor big allocations


7 KVM simple harness
7 KVM
2 KVM + 5 Bochscpu
1 KVM + 6 KVM dirty
thousands of basic blocs covered
time (in hours)
size of corpus (in KB)
executions per seconds
time (in hours)


7 KVM simple harness
7 KVM
2 KVM + 5 Bochscpu
1 KVM + 6 KVM dirty
number of campaigns with crashes
time (in hours)


size of corpus (in KB)


executions per seconds
time (in hours)

Crashes Encountered

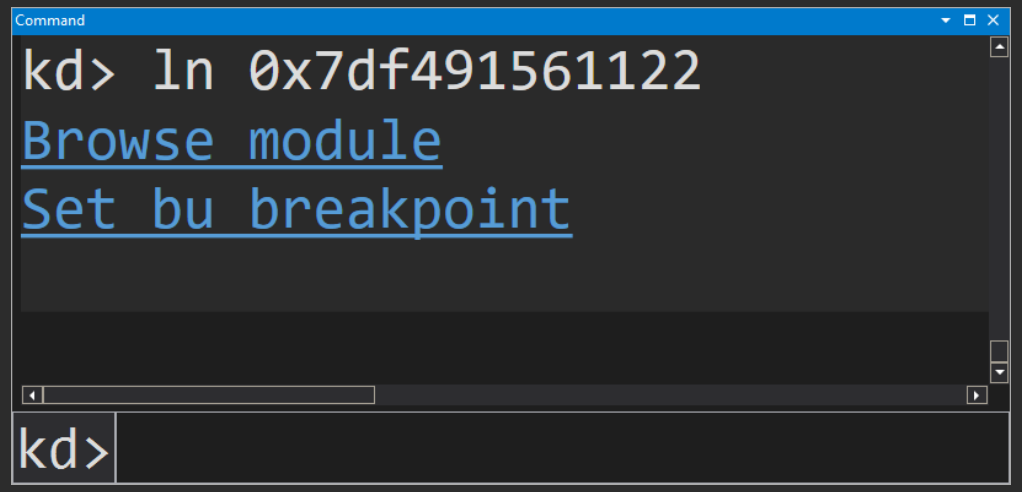
- A **dozen** of crashes!
  - all spurious
  - either linked to an erroneous dump
  - our aggressively optimized harness
  - or our *wtf* modifications

- one seemingly spurious crash harder to analyse
  - `crash-ACCESS_VIOLATION_WRITE-0x7df491561122`

Crash analysis

crash-ACCESS_VIOLATION_WRITE-0x7df491561122

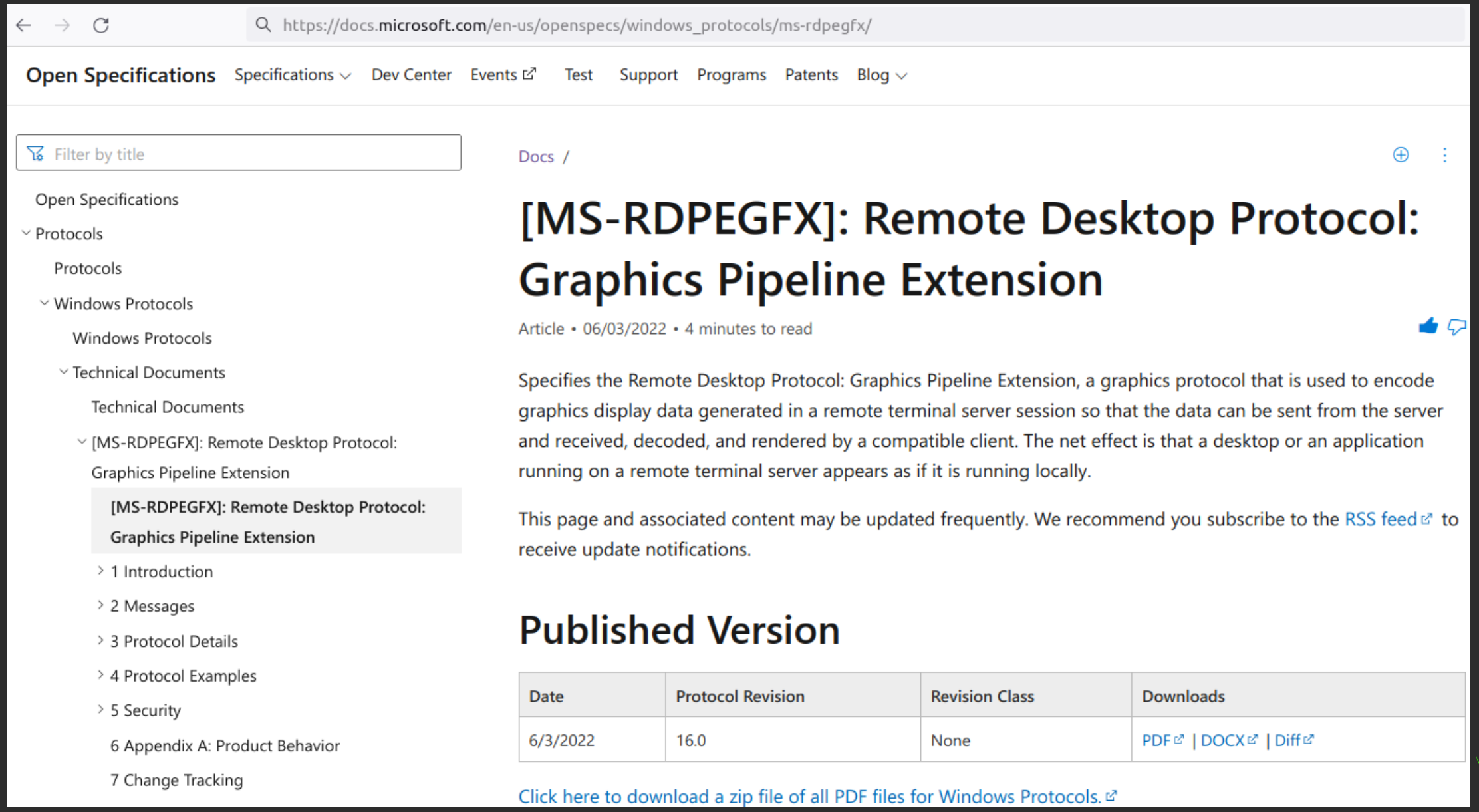
What is at `0x7df491561122`?



- nothing in the dump ?!?

- Reproduce?
  - ✅ Bochs and Kvm
  - ❌ Real system
- Trace
  - Dozens of GigaBytes
  - Symbolizer hangs
  - VSCode hangs

MS-RDPEGFX Open Spec.

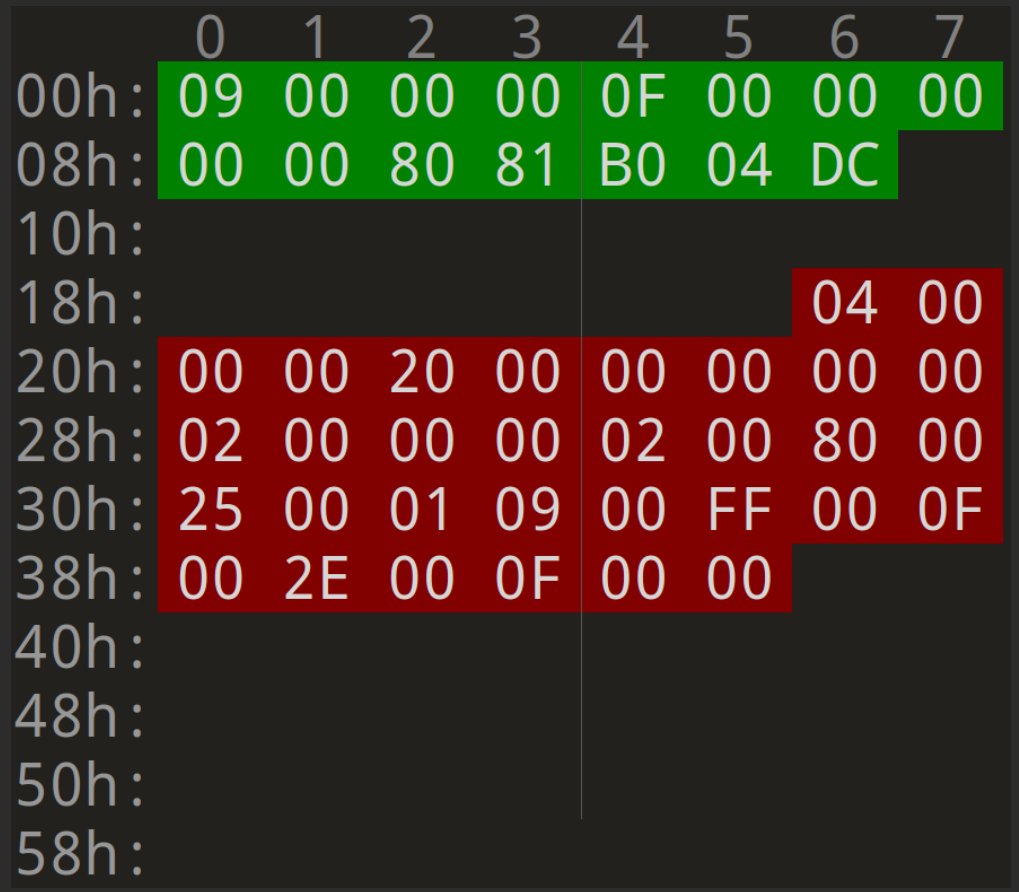


Crash Witness

	0	1	2	3	4	5	6	7
00h:	09	00	00	00	0F	00	00	00
08h:	00	00	80	81	B0	04	DC	09
10h:	00	00	00	0F	00	00	00	00
18h:	00	80	00	00	7B	00	04	00
20h:	00	00	20	00	00	00	00	00
28h:	02	00	00	00	02	00	80	00
30h:	25	00	01	09	00	FF	00	0F
38h:	00	2E	00	0F	00	00	06	00
40h:	00	00	1C	00	00	00	00	00
48h:	00	00	00	00	00	00	15	00
50h:	00	00	00	00	00	00	40	00
58h:	30	00						

- CREATE_SURFACE
- CREATE_SURFACE
- SOLIDFILL
- SURFACE_TO_CACHE

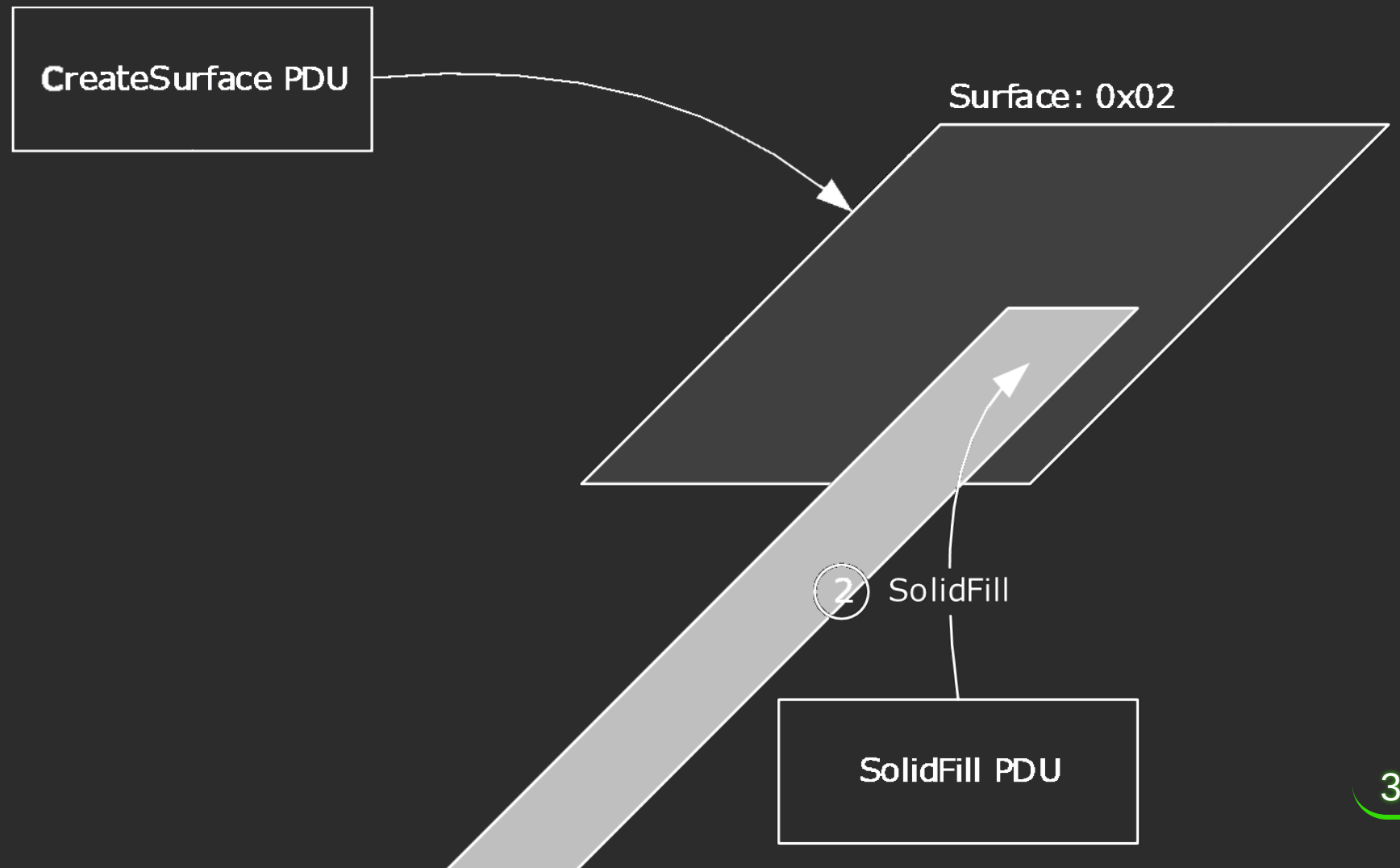
Minimal Crash Witness



```
      0  1  2  3  4  5  6  7
00h: 09 00 00 00 0F 00 00 00
08h: 00 00 80 81 B0 04 DC
10h:
18h:                     04 00
20h: 00 00 20 00 00 00 00 00
28h: 02 00 00 00 02 00 80 00
30h: 25 00 01 09 00 FF 00 0F
38h: 00 2E 00 0F 00 00
40h:
48h:
50h:
58h:
```

- `CREATE_SURFACE`
  - `surfaceId`
  - surface dimensions
- `SOLIDFILL`
  - `surfaceId`
  - fill color
  - areas to fill

MS-RDPEGFX Open Spec.



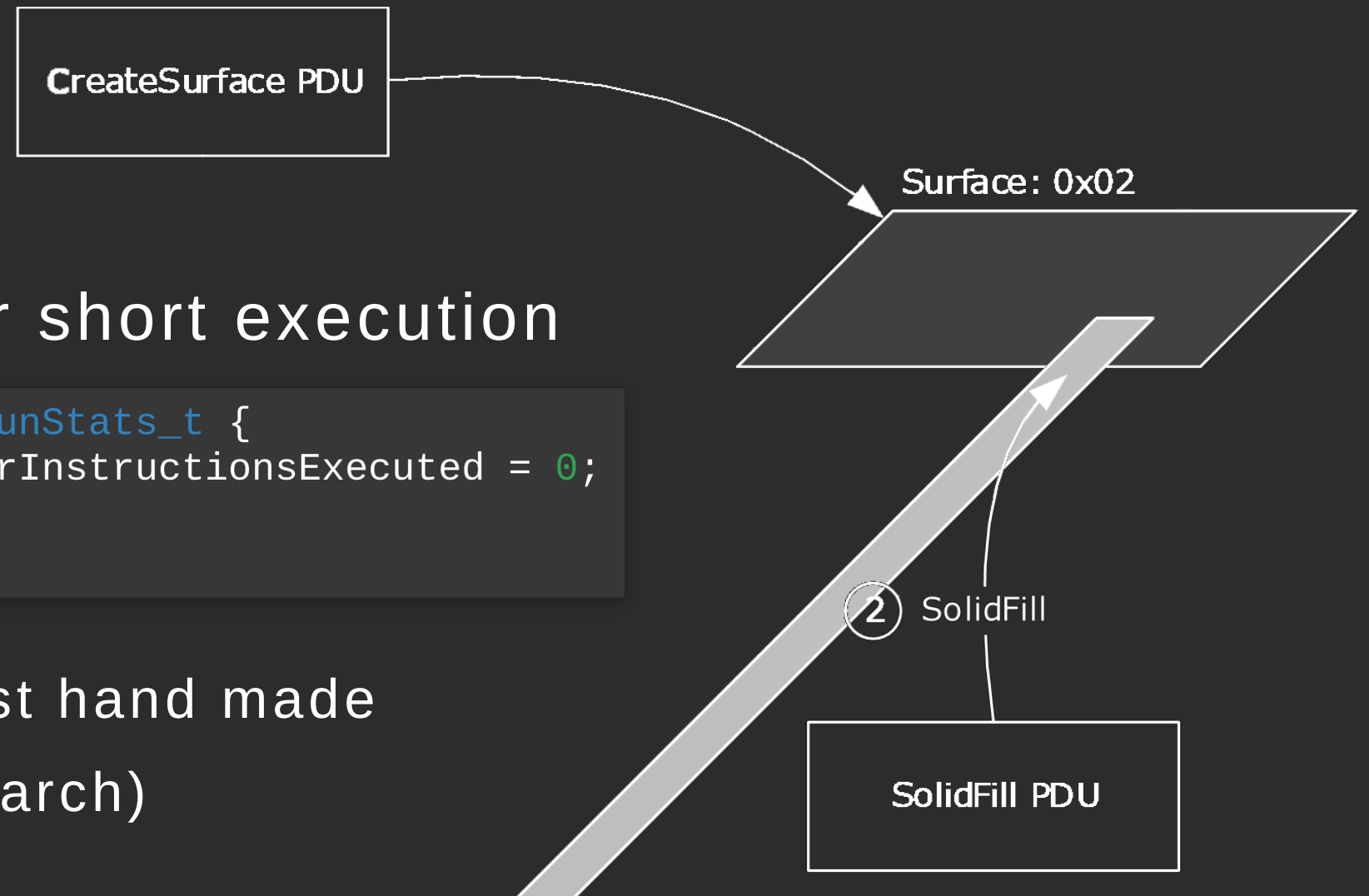
Crash Trace Minimization

Fuzz:

Optimize for short execution

```
struct BochscpuRunStats_t {
  uint64_t NumberInstructionsExecuted = 0;
  // ...
}
```

(Actually... just hand made dichotomic search)



Crash Trace Minimization

💥 Crash 📉 218MB trace

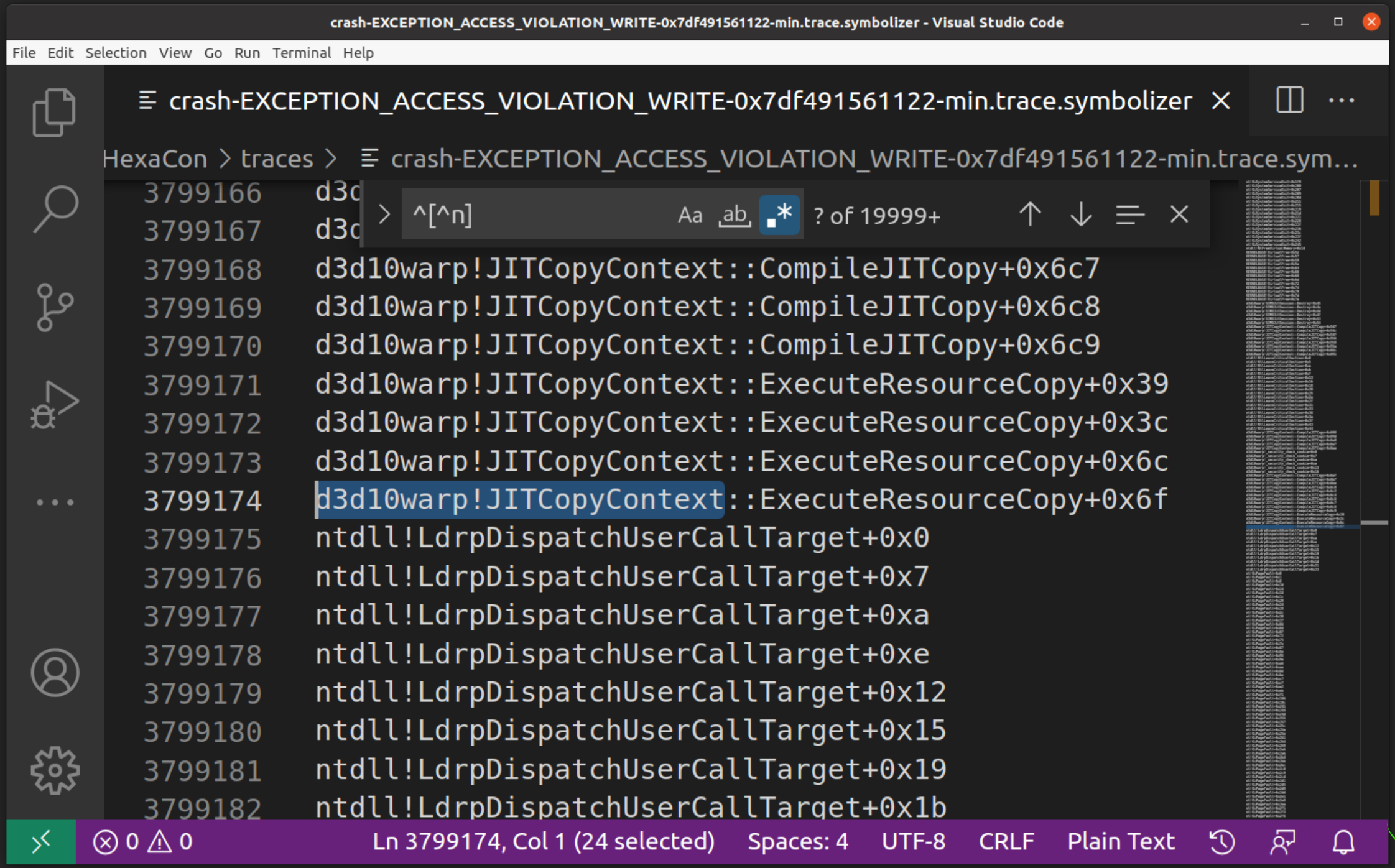
	0	1	2	3	4	5	6	7
00h:	09	00	00	00	0F	00	00	00
08h:	00	00	00	81	00	04	21	04
10h:	00	00	00	18	00	00	00	00
18h:	00	0B	AD	C0	DE	01	00	00
20h:	00	00	00	01	00	02	7F	

- `CREATE_SURFACE_PDU`
  - Width: 0x8100
  - Height: 0x400
- `SOLIDFILL_PDU`
  - Top Left: (0, 0)
  - Bottom Right: (1, **0x7F02**)

👌 No crash

	0	1	2	3	4	5	6	7
00h:	09	00	00	00	0F	00	00	00
08h:	00	00	00	81	00	04	21	04
10h:	00	00	00	18	00	00	00	00
18h:	00	0B	AD	C0	DE	01	00	00
20h:	00	00	00	01	00	01	7F	

- `CREATE_SURFACE_PDU`
  - Width: 0x8100
  - Height: 0x400
- `SOLIDFILL_PDU`
  - Top Left: (0, 0)
  - Bottom Right: (1, **0x7F01**)


crash-EXCEPTION_ACCESS_VIOLATION_WRITE-0x7df491561122-min.trace.symbolizer - Visual Studio Code
File Edit Selection View Go Run Terminal Help
crash-EXCEPTION_ACCESS_VIOLATION_WRITE-0x7df491561122-min.trace.symbolizer
HexaCon > traces > crash-EXCEPTION_ACCESS_VIOLATION_WRITE-0x7df491561122-min.trace.sym...
^[^n]
Aa
ab
.*
? of 19999+
3799166 d3d
3799167 d3d
3799168 d3d10warp!JITCopyContext::CompileJITCopy+0x6c7
3799169 d3d10warp!JITCopyContext::CompileJITCopy+0x6c8
3799170 d3d10warp!JITCopyContext::CompileJITCopy+0x6c9
3799171 d3d10warp!JITCopyContext::ExecuteResourceCopy+0x39
3799172 d3d10warp!JITCopyContext::ExecuteResourceCopy+0x3c
3799173 d3d10warp!JITCopyContext::ExecuteResourceCopy+0x6c
3799174 d3d10warp!JITCopyContext::ExecuteResourceCopy+0x6f
3799175 ntdll!LdrpDispatchUserCallTarget+0x0
3799176 ntdll!LdrpDispatchUserCallTarget+0x7
3799177 ntdll!LdrpDispatchUserCallTarget+0xa
3799178 ntdll!LdrpDispatchUserCallTarget+0xe
3799179 ntdll!LdrpDispatchUserCallTarget+0x12
3799180 ntdll!LdrpDispatchUserCallTarget+0x15
3799181 ntdll!LdrpDispatchUserCallTarget+0x19
3799182 ntdll!LdrpDispatchUserCallTarget+0x1b
0 0
Ln 3799174, Col 1 (24 selected)
Spaces: 4
UTF-8
CRLF
Plain Text

d3d10warp.dll

Windows Advanced Rasterization Platform

> *a high speed, fully conformant software rasterizer*
> [...] *installed on Windows 7* [and up]

- used when HW acc is undesirable or unavailable
  - ✅ Repro. on real system with generic VGA driver

Root Cause Identification by *Differential Trace Analysis*

Fancy way of saying that we compared the crash and no-crash traces...

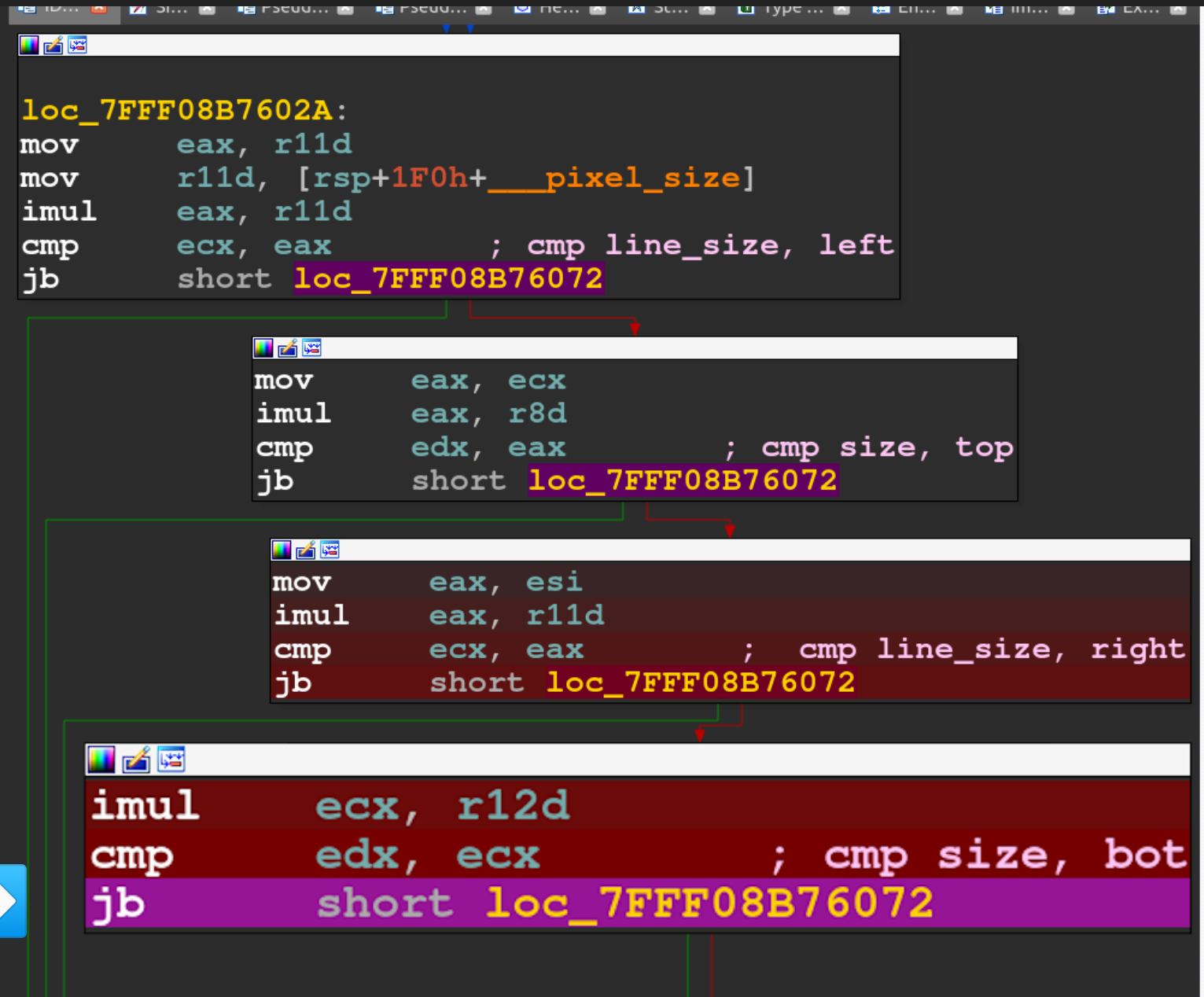
```
3232366  d3d10warp!UMContext::CopyImmediateData+0x2da
3232367  d3d10warp!UMContext::CopyImmediateData+0x2dc
3232368  d3d10warp!UMContext::CopyImmediateData+0x2e0
3232369  d3d10warp!UMContext::CopyImmediateData+0x2e2
3232370  d3d10warp!UMContext::CopyImmediateData+0x2e4
3232371  d3d10warp!UMContext::CopyImmediateData+0x2e6
3232372  d3d10warp!UMContext::CopyImmediateData+0x2ea
3232373  d3d10warp!UMContext::CopyImmediateData+0x2ec
3232374  d3d10warp!UMContext::CopyImmediateData+0x2ee
3232375  d3d10warp!UMContext::CopyImmediateData+0x2f2
3232376  d3d10warp!UMContext::CopyImmediateData+0x2f4
3232377  d3d10warp!UMContext::CopyImmediateData+0x2f6
3232378  d3d10warp!UMContext::CopyImmediateData+0x2fe
3232379  d3d10warp!UMContext::CopyImmediateData+0x301
3232380  d3d10warp!UMContext::CopyImmediateData+0x303
3232381  d3d10warp!UMContext::CopyImmediateData+0x306
3232382  d3d10warp!UMContext::CopyImmediateData+0x30a
3232383  d3d10warp!UMContext::CopyImmediateData+0x30c
3232384  d3d10warp!UMContext::CopyImmediateData+0x1b0
3232385  d3d10warp!UMContext::CopyImmediateData+0x1b7
3232386  d3d10warp!UMContext::CopyImmediateData+0x1bf
3232387  d3d10warp!UMContext::CopyImmediateData+0x1c3
3232388  d3d10warp!UMContext::CopyImmediateData+0x1c7
3232389  d3d10warp!UMContext::CopyImmediateData+0x1cb
3232390  d3d10warp!UMContext::CopyImmediateData+0x1d2
3232391  d3d10warp!UMContext::CopyImmediateData+0x506
3232392  d3d10warp!UMContext::CopyImmediateData+0x50b
3232393  d3d10warp!UMContext::CopyImmediateData+0x52a
```

```
3232366  d3d10warp!UMContext::CopyImmediateData+0x2e2
3232367  d3d10warp!UMContext::CopyImmediateData+0x2e4
3232368  d3d10warp!UMContext::CopyImmediateData+0x2e6
3232369  d3d10warp!UMContext::CopyImmediateData+0x2ea
3232370  d3d10warp!UMContext::CopyImmediateData+0x2ec
3232371  d3d10warp!UMContext::CopyImmediateData+0x2ee
3232372  d3d10warp!UMContext::CopyImmediateData+0x2f2
3232373  d3d10warp!UMContext::CopyImmediateData+0x2f4
3232374  d3d10warp!UMContext::CopyImmediateData+0x312
3232375  d3d10warp!UMContext::CopyImmediateData+0x317
3232376  d3d10warp!GetCurrentAddress+0x0
3232377  d3d10warp!GetCurrentAddress+0x4
3232378  d3d10warp!UMContext::CopyImmediateData+0x31c
3232379  d3d10warp!UMContext::CopyImmediateData+0x31f
3232380  d3d10warp!UMContext::CopyImmediateData+0x325
3232381  d3d10warp!UMContext::CopyImmediateData+0x327
3232382  d3d10warp!WarpPlatform::RecordError+0x0
3232383  d3d10warp!WarpPlatform::RecordError+0x5
3232384  d3d10warp!WarpPlatform::RecordError+0xa
3232385  d3d10warp!WarpPlatform::RecordError+0xb
3232386  d3d10warp!WarpPlatform::RecordError+0xf
3232387  d3d10warp!WarpPlatform::RecordError+0x11
3232388  d3d10warp!WarpPlatform::RecordError+0x14
3232389  d3d10warp!WarpPlatform::RecordError+0x1b
3232390  d3d10warp!WarpPlatform::RecordError+0x1e
3232391  ntdll!RtlEnterCriticalSection+0x0
3232392  ntdll!RtlEnterCriticalSection+0x13
3232393  ntdll!RtlEnterCriticalSection+0x16
```

⚠ Cannot compare files because one file is too large.

45

Root Cause Analysis with Tenet



💥 Crash

skip branch

CPU Registers

RCX	0000000000020400
RDX	0000000008100000
R12	000000000007F02
RIP	00007FFF08B76054

👌 No crash

take branch

CPU Registers

RCX	0000000000020400
RDX	0000000008100000
R12	000000000007F01
RIP	00007FFF08B76054

Root Cause Analysis

```
loc_7FFF08B7602A:
mov     eax, r11d
mov     r11d, [rsp+1F0h+___pixel_size]
imul    eax, r11d
cmp     ecx, eax          ; cmp line_size, left
jb      short loc_7FFF08B76072
```

```
mov     eax, ecx
imul    eax, r8d
cmp     edx, eax          ; cmp size, top
jb      short loc_7FFF08B76072
```

```
mov     eax, esi
imul    eax, r11d
cmp     ecx, eax          ;  cmp line_size, right
jb      short loc_7FFF08B76072
```

```
imul    ecx, r12d
cmp     edx, ecx          ; cmp size, bot
jb      short loc_7FFF08B76072
```

💥 Crash

edx > 0x800

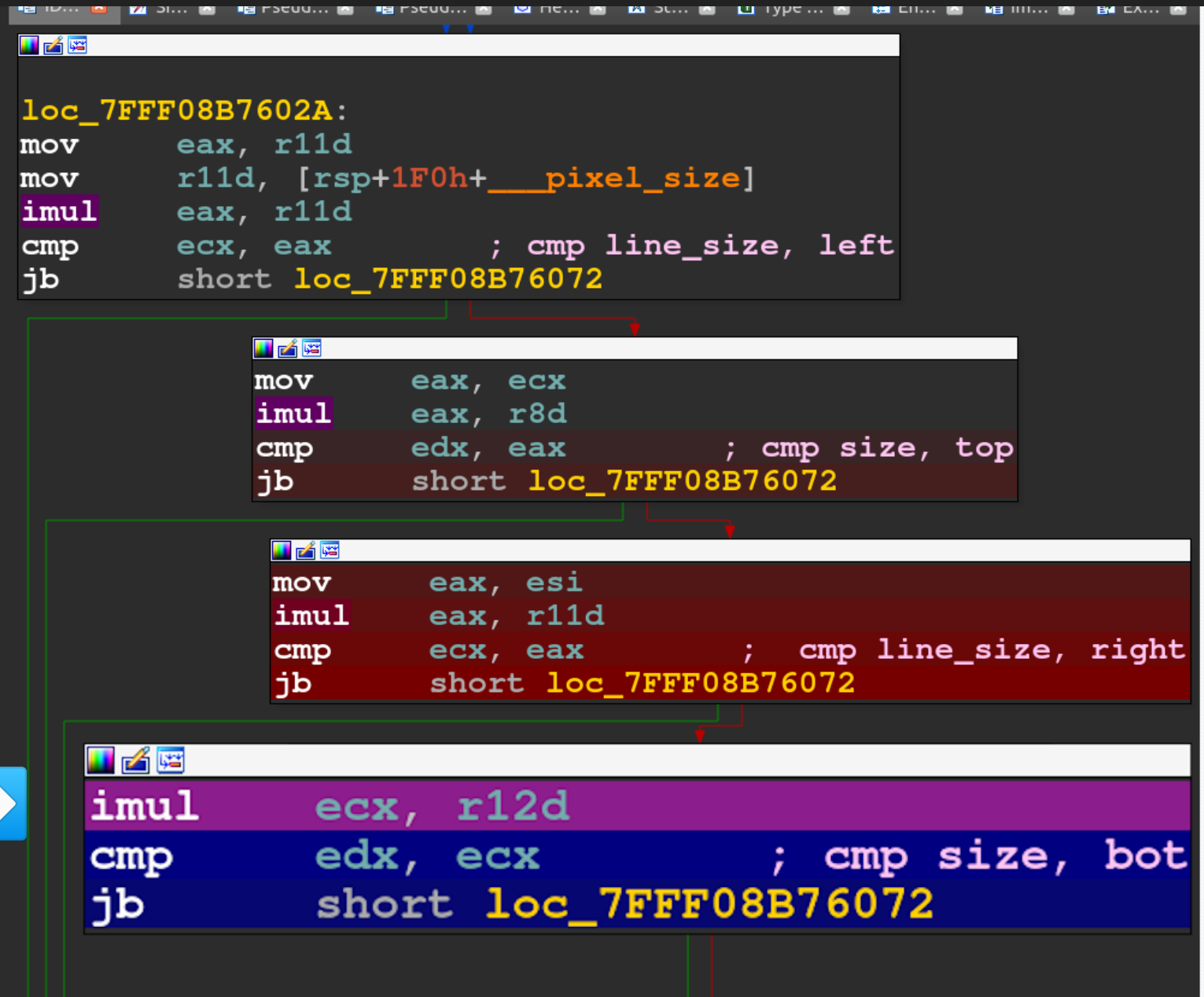
RCX	0000000000000800
RDX	0000000008100000
R12	000000000007F02
RIP	00007FFF08B76052

👌 No crash

edx < 0xFFFE400

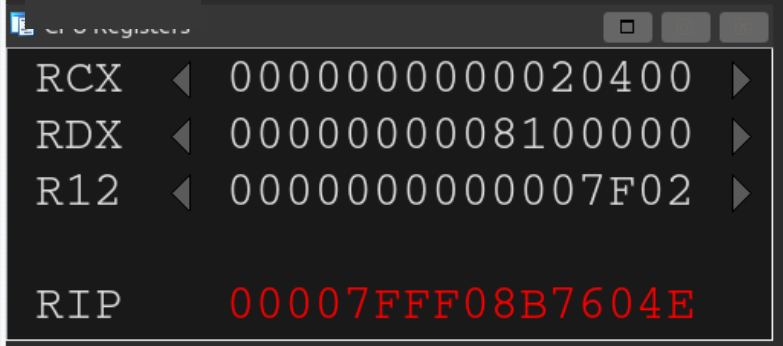
RCX	00000000FFFE0400
RDX	0000000008100000
R12	000000000007F01
RIP	00007FFF08B76052

Root Cause Analysis



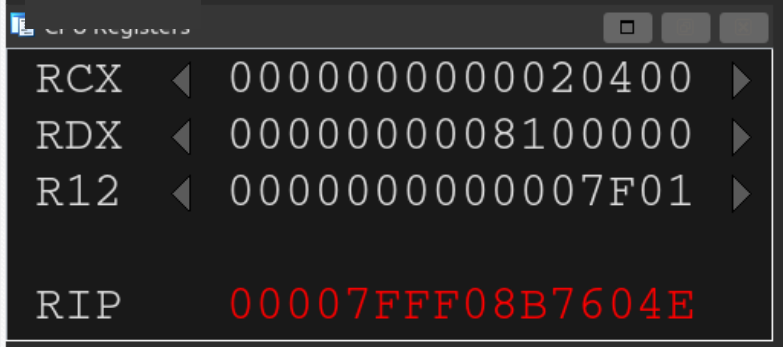
💥 Crash

R12 = 0x7f02



👌 No crash

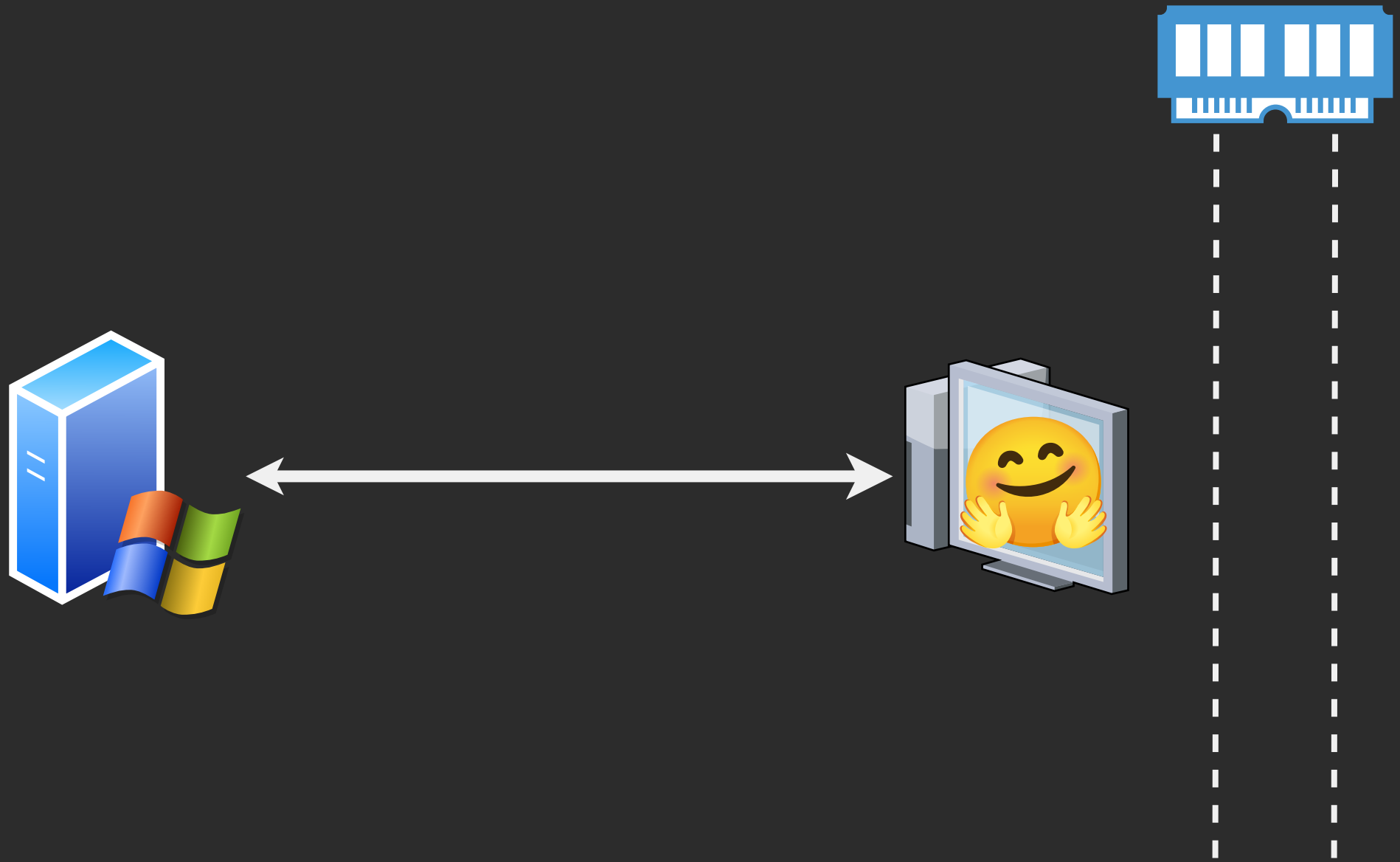
R12 = 0x7f01



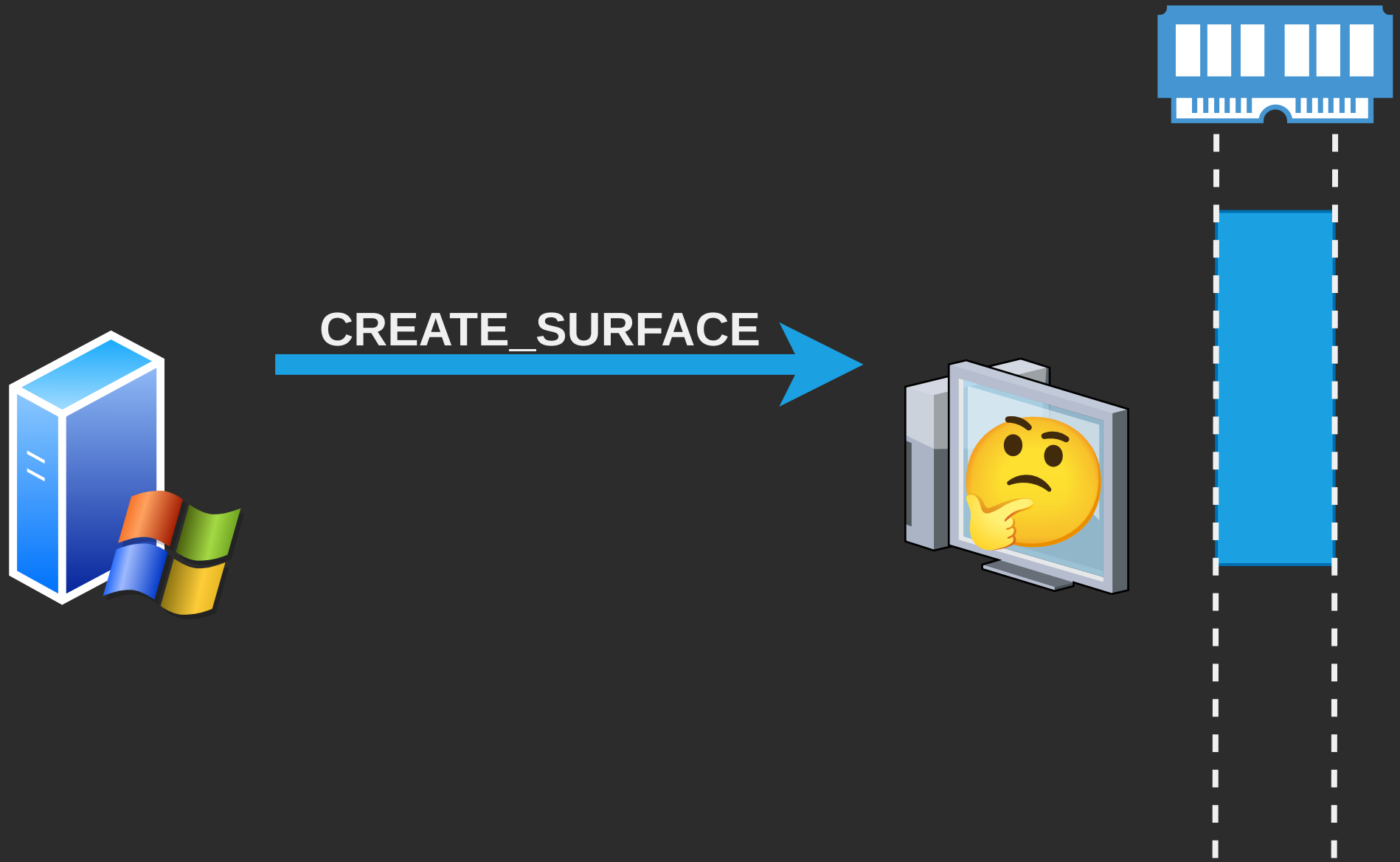
Crash Analysis - Recap

- 🤨 Unvalidated crash with huge trace
- 📖 Seems legitimate after reading specification
- 🪓 crash trace minimization
  - oobw in d3d10warp.dll ➡️ Repro. without HW 3D
- 🖱️ root cause identification with **diff**
- 🔬 root cause analysis with Tenet by @gaasedelen
  - integer overflow in `UMContext::CopyImmediateData`

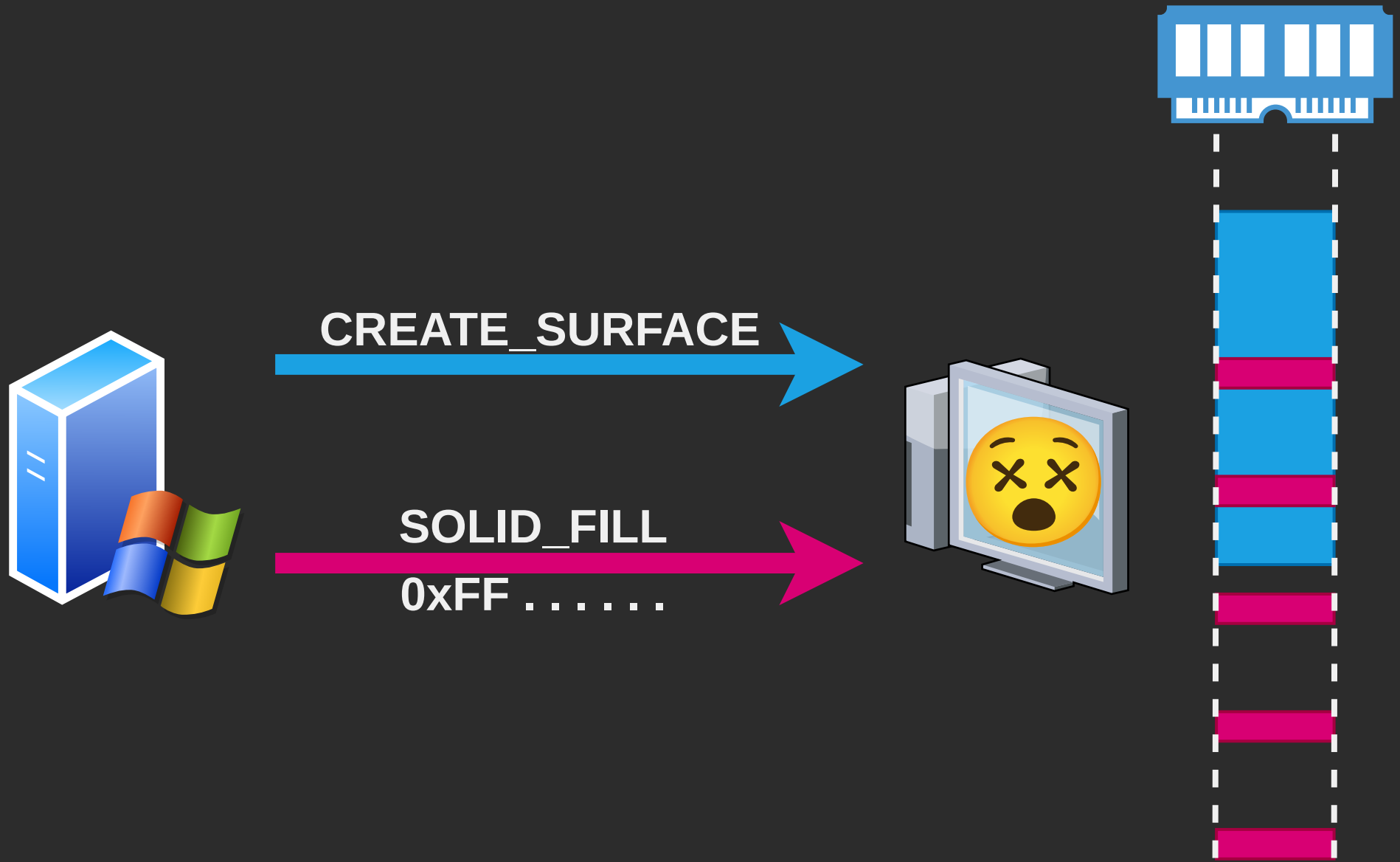
Impact



Impact

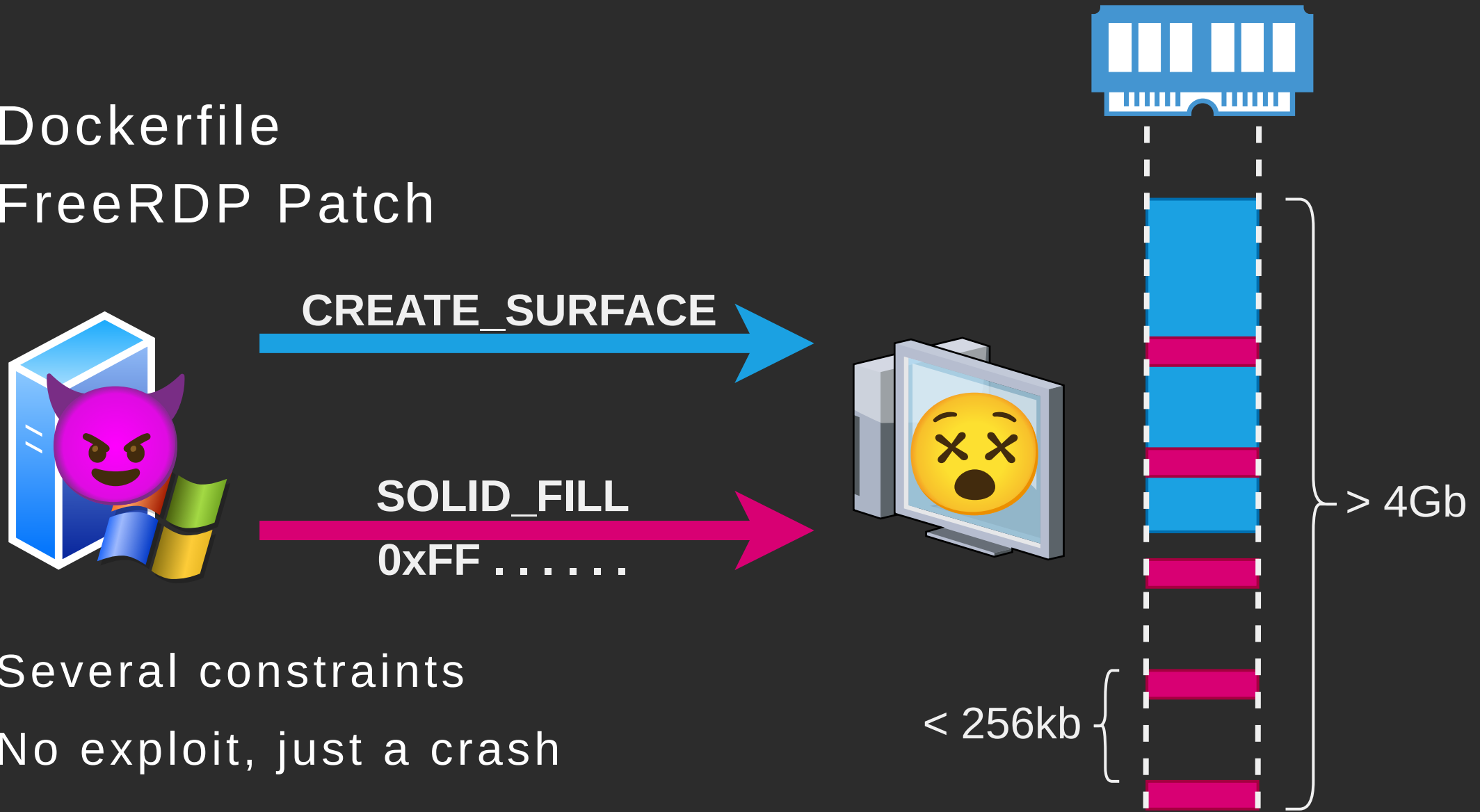


Impact



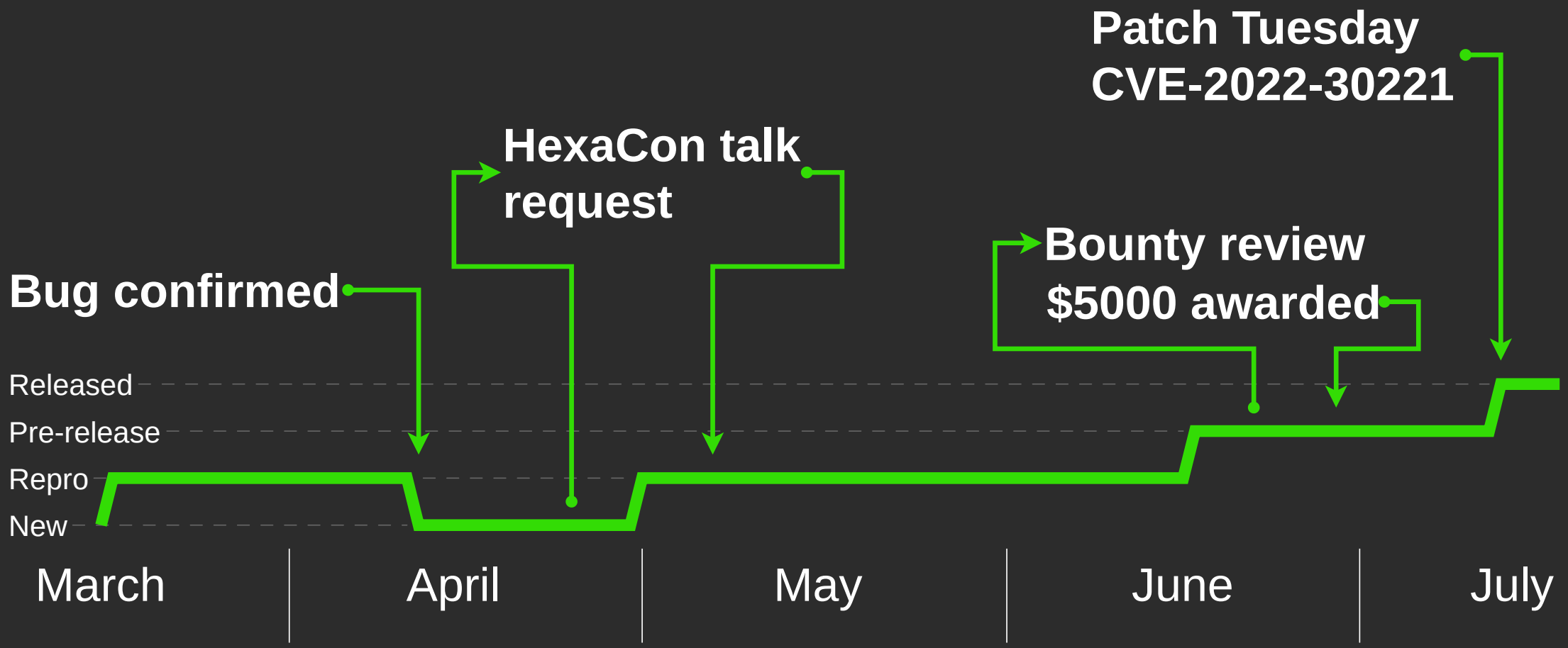
PoVuln: Malicious FreeRDP Server

- Dockerfile
- FreeRDP Patch



- Several constraints
- No exploit, just a crash

Responsible disclosure



Conclusion

OOB write in d3d10warp
triggerable remotely through RDPEGFX

- On the fly modification of fuzzing data
- Exotic coverage
- Trace minimization

- *wtf*, *diff*, and *tenet*
- MS Open Spec. Program
- MS Win. Protocol Test Suites

more on thalium.re

d3d10warp.dll Broader Impact?

- without hardware acceleration, loaded by:
  - explorer.exe
  - msedge.exe and msedgewebview2.exe
  - winword.exe and other MS Office executables
  - ...
- *with* hardware acceleration, loaded by:
  - explorer.exe and a few others

Anyway: CVE-2022-30221 fixed in July -- No exploit

THALIUM

Fuzzing RDPEGFX with *what the fuzz*

🤗 Questions?

/HEXACON/