A very quick look inside win32k_rs

Cindy Xiao

REcon 2023 Lightning Talk

whoami

Security Researcher @ [REDACTED]

- (this is just me poking around in my free time though)
- Malware Reverse Engineering, Threat Intelligence
- Former embedded systems / C++ dev
- With regards to Win32k, I have no idea what I’m doing

CindA Xiao
@cxiao@infosec.exchange

In the new Rust Windows kernel GDI code, there is a new global allocator registered named `gdi_alloc::Win32Allocator`. It calls `Win32AllocPool` with a fun new pool tag name, "Rust"!

#rust #rustlang #windows #microsoft #reversing #reverseengineering

```
loc::Win32Allocator as core::alloc::global::GlobalAlloc>::alloc::hbe84ca87c2f258d5(int64_t arg1, uint64_t layout_
align u> 0x10) {
anicking::panic::haf36b911569154db(expr: "assertion failed: layout.align()…"  )
n
ALT ol(nBytes: layout_size, tag: 'Rust') __tailcall
```

May 14, 2023 at 4:17:39 AM **(Edited)** · · Elk

1 4 9

CindA Xiao @cxiao@infosec.exchange 3w

For the specific GDI objects, there are still allocations made with the existing GDI-specific pool tags.

It looks like the `rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>` object uses the existing GDI pool tag `Gscn` (i.e. `GDITAG_SCAN_ARRAY`) for vector allocations. (Probably `gdi_alloc::TaggedAllocator<_>` requires specifying a pool tag)

I also see `Gedg` (i.e. `GDITAG_EDGE`) being used in `gdi_rust::region::from_path::GlobalEdgeTable::add_edge`, and `gdi_rust::region::from_path::ActiveEdgeTable::new`, among other places.

#rust #rustlang #windows #microsoft #reversing #reverseengineering

1 1

CindA Xiao @cxiao@infosec.exchange 3w

The Rust Windows kernel GDI code also has symbols for `fallible_vec::FallibleVec<T,A>`, which looks like a non-panicking `Vec` implementation. `try_extend`, `try_extend_from_slice`, `try_splice_in`, and `try_insert` are all implemented.

In fact it looks suspiciously similar to the `rust_fallible_vec` crate, which Microsoft recently open-sourced: github.com/microsoft/rust_fall... 😂

<- Mastodon thread with writeup

infosec.exchange/@cxiao

cxiao.net

github.com/cxiao

https://infosec.exchange/@cxiao/110366048880535679

https://infosec.exchange/@cxiao/110360594370994764

Mark Russinovich
@markrussinovich

If you're on the Win11 Insider ring, you're getting the first taste of Rust in the Windows kernel!

```
C:\Windows\System32>dir win32k*
 Volume in drive C has no label.
 Volume Serial Number is E60B-9A9E

 Directory of C:\Windows\System32

04/15/2023  09:50 PM           708,608 win32k.sys
04/15/2023  09:49 PM         3,424,256 win32kbase.sys
04/15/2023  09:49 PM           110,592 win32kbase_rs.sys
04/15/2023  09:50 PM         4,194,304 win32kfull.sys
04/15/2023  09:49 PM            40,960 win32kfull_rs.sys
04/15/2023  09:49 PM            69,632 win32kns.sys
04/15/2023  09:49 PM            98,304 win32ksgd.sys
               7 File(s)      8,646,656 bytes
               0 Dir(s)  116,366,049,280 bytes free
```

_rs = Rust!

5:50 PM · May 10, 2023 · **692.2K** Views

Windows 11: The journey to security by-default

David “dwizzzle” Weston
Director of OS Security
@dwizzzleMSFT

BlueHat IL 2023

Windows 11: The journey to security by-default

David "dwizzzle" Weston
Director of OS Security
@dwizzzleMSFT

Win32k GDI port to Rust

- Ported the REGION data type and functions

Region Operations

Article • 01/07/2021 • 3 contributors

Feedback

Applications can combine regions, compare them, paint or invert their interiors, draw a frame around them, retrieve their dimensions, and test whether the cursor lies within their boundaries.

Additional resources

Documentation

Bitmap Classifications - Win32 apps

Bitmap Classifications

RGNDATA (wingdi.h) - Win32 apps

The RGNDATA structure contains a header and an array of rectangles that compose a region. The rectangles are sorted top to...

MaskBlt function (wingdi.h) - Win32 apps

The MaskBlt function combines the color data for the source and destination bitmaps using the specified mask and...

Show 5 more

Setup: Getting the Binaries

Binaries from Windows 11 Insider Preview **25357.1** (zn_release) amd64

Option 1: Update package (UUP) -> ISO

uupdump.net

gus33000/UUPMediaCreator

UUP dump

Menu

Browse known builds

Windows 11 Insider Preview 25357.1 (zn_release) amd

Sort results by addition date

1 builds were found for your query.

Build	Architecture	Date added
Windows 11 Insider Preview 25357.1 (zn_release) amd64	x64	2023-05-04 17:00:13 UTC

Setup: Getting the Binaries

Binaries from Windows 11 Insider Preview **25357.1** (zn_release) amd64

Option 2: Directly from the MS Symbol Servers

msdl.microsoft.com/download/symbols/win32kbase_rs.sys/272C4A031b000/win32kbase_rs.sys

msdl.microsoft.com/download/symbols/win32kfull_rs.sys/8264C482a000/win32kfull_rs.sys

Not in Winbindex yet):

CindA Xiao
@cxiao@infosec.exchange

For the new Windows kernel Rust GDI stuff that is all the rage now (win32kbase_rs.sys, win32kfull_rs.sys): here are the links to download copies of those binaries, from the Microsoft Symbol Server:

msdl.microsoft.com/download/sy...

msdl.microsoft.com/download/sy...

These should be the versions that are in Windows 11 Insider Preview 25357.1 (zn_release) amd64 . The SHA-256 hashes are:

87ee0235caf2c97384581e74e525756794fa91b666eaacc955fc7859f540430d win32kbase_rs.sys
2efb9ea4032b3dfe7bf7698bd35e3ea3817d52f4d9a063b966f408e196957208 win32kfull_rs.sys

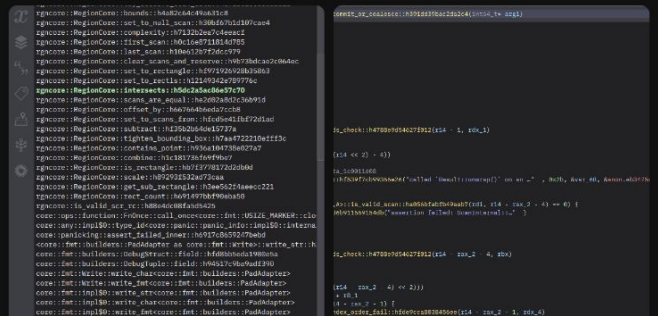
(I first extracted these files myself from the update package for build 25357.1, then generated the symbol server download URLs from the PE metadata in the files)

Of course, in addition to the actual executables, symbols are available from the symbol server as well (see screenshot).

@analog_feelings already did some reversing of win32kbase_rs.sys several weeks ago, here: tech.lgbt/@analog_feelings/110... 🤘

Now, time for me to go figure out how to actually reverse Rust 🦀

#rust #rustlang #windows #reversing #reverseengineering #microsoft



<- Mastodon post with links to symbol server downloads

https://infosec.exchange/@cxiao/110360594370994764

Setup: Symbols

Symbols are available from the Microsoft Symbol Server.

```
[Default] Check file exists: https://msdl.microsoft.com/download/symbols/win32kbase_rs.pdb/
55CC41388DDB1E80EC7722B9AF8F38D01/win32kbase_rs.pdb
[Default] HEAD: https://msdl.microsoft.com/download/symbols/win32kbase_rs.pdb/
55CC41388DDB1E80EC7722B9AF8F38D01/win32kbase_rs.pdb
[Default] Read file: https://msdl.microsoft.com/download/symbols/win32kbase_rs.pdb/
55CC41388DDB1E80EC7722B9AF8F38D01/win32kbase_rs.pdb
[Default] GET: https://msdl.microsoft.com/download/symbols/win32kbase_rs.pdb/
55CC41388DDB1E80EC7722B9AF8F38D01/win32kbase_rs.pdb
```

```
_ZN7rgncore4scan20ScanBuilder$LT$A$GT$11append_wall17h9b4944cd350dd5f1E
_ZN7rgncore4scan20ScanBuilder$LT$A$GT$18commit_or_coalesce17h391dd39bac2da2c4E
_ZN7rgncore4scan54ScanBuilder$LT$gdi_alloc..TaggedAllocator$LT$_$GT$$GT$3new17hec32907465930b8aE
_ZN7rgncore4scan54ScanBuilder$LT$gdi_alloc..TaggedAllocator$LT$_$GT$$GT$11append_scan17h6aede44f01b7bc6fE
_ZN5alloc7raw_vec11finish_grow17hfdc9d17557afcdd7E.llvm.16677895477366079116
_ZN5alloc7raw_vec11finish_grow17h84acf38cd5444a9aE.llvm.16677895477366079116
.weak.memset.default.__llvm_retpoline_r11
.weak.memcmp.default.__llvm_retpoline_r11
_ZN12gdi_bindings5point5POINT3new17ha934527bfa81e1ffE
_ZN[illegible]7_$LT$alloc..vec..Vec$LT$T$C$A$GT$$u20$as$u20$fallible_vec..FallibleVec$LT$T$C$A$GT$$GT$10try_extend1
_Z[illegible]37_$LT$alloc..vec..Vec$LT$T$C$A$GT$$u20$as$u20$fallible_vec..FallibleVec$LT$T$C$A$GT$$GT$10try_extend1
_[illegible]37_$LT$alloc..vec..Vec$LT$T$C$A$GT$$u20$as$u20$fallible_vec..FallibleVec$LT$T$C$A$GT$$GT$13try_splice_
_[illegible]37_$LT$alloc..vec..Vec$LT$T$C$A$GT$$u20$as$u20$fallible_vec..FallibleVec$LT$T$C$A$GT$$GT$21try_extend_
```

```
rgncore::scan::ScanBuilder<A>::append_wall::h9b4944cd350dd5f1
rgncore::scan::ScanBuilder<A>::commit_or_coalesce::h391dd39bac2da2c4
rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>::new::hec32907465930b8a
rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>::append_scan::h6aede44f01b7bc6f
alloc::raw_vec::finish_grow::ha096a394c3159422
alloc::raw_vec::finish_grow::h84acf38cd5444a9a
.weak.memset.default.__llvm_retpoline_r11
memcmp
gdi_bindings::point::POINT::new::ha934527bfa81e1ff
<alloc::vec::Vec<T,A> as fallible_vec::FallibleVec<T,A>>::try_extend::haf171a9fcdacb471
<alloc::vec::Vec<T,A> as fallible_vec::FallibleVec<T,A>>::try_extend::he10c6c17e44c2ad3
<alloc::vec::Vec<T,A> as fallible_vec::FallibleVec<T,A>>::try_splice_in::h20cbb10bb5234f51
<alloc::vec::Vec<T,A> as fallible_vec::FallibleVec<T,A>>::try_extend_from_slice::h7345911767d1aeb0
```

Setup: Symbols

Demangle with:

- IDA: cxiao/ida-rust-untangler
- Binary Ninja: inspier/BinjaRustDemangler
  - (search for “Rust Demangler” in the built-in plugin manager)
- Ghidra: ???

Allocation

```
rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>::new::hec32907465930b8a
rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>::append_scan::h6aede44f01b7bc6f
alloc::raw_vec::finish_grow<gdi_alloc::TaggedAllocator<Gscn>>
alloc::raw_vec::finish_grow<gdi_alloc::Win32Allocator>
alloc::raw_vec::finish_grow<gdi_alloc::TaggedAllocator<Gedg>>::h31354883a0fcda7b
<gdi_alloc::Win32Allocator as core::alloc::global::GlobalAlloc>::alloc::hbe84ca87c2f258d5
<gdi_alloc::Win32Allocator as core::alloc::global::GlobalAlloc>::dealloc::h8ec22908e6d1164e
```

gdi_alloc::Win32Allocator

```
int64_t <gdi_alloc::Win32Allocator as core::alloc::global::GlobalAlloc>::alloc::hbe84ca87c2f258d5(
    int64_t arg1, uint64_t layout_size, int64_t layout_align)

{
    if (layout_align > MEMORY_ALLOCATION_ALIGNMENT
    {
        core::panicking::panic::haf36b911569154db("assertion failed: layout.align()…"  , 0x3f, &data_1c0012028);
        /* no return */
    }
    /* tailcall */
    return Win32AllocPool(layout_size, 'Rust');
}
```

gdi_alloc::TaggedAllocator<_>>

```
rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>::new::hec32907465930b8a
rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>::append_scan::h6aede44f01b7bc6f
```

```
if (alloc_alignment == 0)
{
    new_layout->layout.size = alloc_size;
    new_layout->layout.align = 0;
ERR_EXIT:
    result_err = 1;
}
else
{
    int64_t* allocated_1;
    if (current_memory->layout.align == 0)
    {
        if (alloc_alignment > MEMORY_ALLOCATION_ALIGNMENT
        {
        ERR_CLEANUP:
            new_layout->layout.size = alloc_size;
            new_layout->layout.align = alloc_alignment;
            goto ERR_EXIT;
        }
        int64_t* allocated_0 = Win32AllocPool(alloc_size, 'Gscn');
        allocated_1 = allocated_0;
        if (allocated_0 == 0)
        {
            goto ERR_CLEANUP;
        }
    }
```

```
    if (arg3 == 0)
    {
        arg1[1] = arg2;
        arg1[2] = 0;
ERR_EXIT:
        rax_3 = 1;
    }
    else
    {
        int64_t r14_1;
        if (arg4[2] == 0)
        {
            if (arg3 > MEMORY_ALLOCATION_ALIGNMENT
            {
            ERR_CLEANUP:
                arg1[1] = arg2;
                arg1[2] = arg3;
                goto ERR_EXIT;
            }
            int64_t rax_2 = Win32AllocPool(arg2, 'Gedg');
            r14_1 = rax_2;
            if (rax_2 == 0)
            {
                goto ERR_CLEANUP;
            }
        }
        else
```

gdi_alloc::TaggedAllocator<_>>

```
alloc::raw_vec::finish_grow::ha096a394c3159422
alloc::raw_vec::finish_grow::h84acf38cd5444a9a
alloc::raw_vec::finish_grow::h31354883a0fcda7b
```

```
alloc::raw_vec::finish_grow<gdi_alloc::TaggedAllocator<Gscn>>
alloc::raw_vec::finish_grow<gdi_alloc::Win32Allocator>
alloc::raw_vec::finish_grow<gdi_alloc::TaggedAllocator<Gedg>>
```

fallible_vec

```
char const data_1c0012600[0x40b] = "attempt to divide with overfl"
    "owassertion failed: !self.ppath.is_null()gdi_rust\\src\\path"
    "obj.rsd:\\os\\src\\onecoreuap\\windows\\core\\ntgdi\\rust\\r"
    "gncore\\src\\scan.rsassertion failed: index ≤ scan_data.len"
    "()called `Result::unwrap()` on an `Err` valueassertion faile"
    "d: ScanInternal::is_valid_scan(&&*self.scan_data, previous_i"
    "ndex)assertion failed: *wall_count > 0assertion failed: inde"
    "x < self.wall_count()WindowsErrorrgncore\\src\\scan.rscalled"
    " `Result::unwrap()` on an `Err` valueassertion failed: Self:"
    ":is_valid_scan(&scan_data, 0)called `Option::unwrap()` on a "
    "`None` valueassertion failed: index < self.wall_count()asser"
    "tion failed: *wall_count - count as i32 ≥ 0assertion failed"
    ": ScanInternal::is_valid_scan(&&*self.scan_data, previous_in"
    "dex)assertion failed: index ≤ scan_data.len()/rustc/adb13e8"
    "0e8260ed3718296803f35ca166c4293be\\library\\alloc\\src\\vec"
    "\\mod.rsD:\\os\\osdep\\rust.crates-io\\fallible_vec-0.1.0\\s"
    "rc\\lib.rscalled `Option::unwrap()` on a `None` value/rustc/"
    "adb13e80e8260ed3718296803f35ca166c4293be\\library\\core\\src"
    "\\slice\\sort.rs", 0
```

fallible_vec

microsoft/rust_fallible_vec

Open-source implementation of Rust's Vec data type, by Microsoft

Handles failed memory allocations gracefully - leaves the Vec in a consistent state

- Has driven changes upstream in Rust
  - More try_ methods for Vec that don't panic on OOM: https://github.com/rust-lang/rust/pull/95051

```
// ID haf171a9fcdacb471
1c000d3d0  uint64_t
1c000d3d0      _<alloc::vec::Vec<T,gdi_alloc::Win32Allocator> as fallible_vec::Fallib
1c000d3d0      (struct alloc::vec::Vec<T,A>* self, int64_t arg2, int64_t iter_limit)

1c000d3d0  {
1c000d3e3      int64_t rdi = arg2;
1c000d3ef      uint64_t tmp = ((iter_limit - arg2) >> 4);
1c000d3f3      uint64_t raw_vec_cap = self->buf.cap;
1c000d3f7      uint64_t vec_len = self->len;
1c000d404      struct Tuple<u8*, Layout> current_memory;
1c000d404      struct Result<Layout, LayoutError> new_layout;
1c000d404      uint64_t temp0_1;
1c000d404      if ((raw_vec_cap - vec_len) < tmp)
1c000d401      {
1c000d40a          temp0_1 = tmp;  // try_reserve implementation begins here
1c000d40a          tmp = (tmp + vec_len);
1c000d40d          if ((temp0_1 + vec_len) ≥ temp0_1)
1c000d40a          {  // try_reserve implementation begins here
1c000d419              // grow_amortized implementation
1c000d419              uint64_t double_current_cap = (raw_vec_cap + raw_vec_cap);
1c000d41f              if (double_current_cap > tmp)
1c000d41c              {
1c000d41f                  tmp = double_current_cap;
1c000d41f              }
1c000d427              uint64_t layout_size = alloc::raw_vec::MIN_NON_ZERO_CAP;
1c000d42c              if (tmp ≥ 5)  // alloc::raw_vec::MIN_NON_ZERO_CAP + 1
```

Panicking

```
core::any::impl$0::type_id<core::panic::panic_info::impl$0::in
core::panicking::assert_failed_inner::h6917c8659247bebd
core::panicking::panic_fmt::hd60a775b92204b91
core::panicking::panic::haf36b911569154db
core::panicking::panic_bounds_check::h4788e9d54627f012
core::panicking::assert_failed::h1adf1040b605f158
core::panicking::assert_failed::h685b5b86314fac76
```

Global panic handler

rust_begin_unwind

```
1c000ef70  int64_t rust_begin_unwind(int64_t* msg, char* file, uint32_t line) __noreturn

1c000ef70  {
1c000ef83      seh_unwind::implementation::raise_exception::hc52a1220c03bdc19(STATUS_FAIL_FAST_
1c000ef83      /* no return */
1c000ef83  }
```

```
int64_t seh_unwind::implementation::raise_exception::hc52a1220c03bdc19(int32_t exception_code,
    char* exception_message, int64_t exception_NumberParameters) __noreturn

{
    int32_t var_ac = exception_code;
    int32_t status_code_ = win_defs::ntstatus::NTSTATUSError::status::h71e417d20a1a2c45(&var_ac);
    struct EXCEPTION_RECORD exception_record;
    __builtin_memset(&exception_record.ExceptionInformation, 0, 0x78);
    exception_record.ExceptionCode = status_code_;
    exception_record.ExceptionFlags = 0;
    exception_record.ExceptionRecord = 0;
    exception_record.ExceptionAddress = 0;
    *(uint32_t*)((char*)exception_record.ExceptionAddress)[4] = 0;
    exception_record.NumberParameters = exception_NumberParameters;
    int64_t* exception_info_ptr;
    int64_t exception_info_len;
    exception_info_ptr = core::slice::index::impl$4::index_mut<usize>(exception_NumberParameters, &exception_reco
    core::slice::<impl [T]>::copy_from_slice::h17325824e865690e(exception_info_ptr, exception_info_len, exception
    RtlRaiseException(&exception_record);
    trap(6);  // ud2 instruction (invalid opcode panic)
}
```

Enabling the feature flag

Shoutout to my Windows Internals training classmate, Brent

vivetool: [thebookisclosed/ViVe]

```
vivetool /enable /id:37356106
```

(Rust_GDI_REGION feature flag)

Win32k GDI port to Rust

- Currently disabled via a feature-flag.

Visiting Vibranium Velocity

A look at new changes to everyone's favorite A/B system

Albacore · Follow

4 min read · Sep 19, 2019

Whether you are a developer or an end user taking part in the Windows Insider program, we can probably agree on one thing: **Velocity**, the mechanism Microsoft uses to A/B test features in Windows 10, is interesting to tinker with.

First, a tiny bit of history

Even though public 3rd party tooling for interacting with Velocity has been released in April 2018, its existence dates back to at least February 2016. The fact that it's taken roughly 2 years for people to notice and reverse engineer what goes on behind the scenes proves that the project has achieved its goals. Besides catering to many teams' needs and simplifying the internal testing of new breaking changes, it also provided a fair bit of challenge for

Enabling the feature flag

```
long LoadAndConnectRustCode()

    data_1c02ad510 = ?gxsGlobals@@3UtagXSGLOBALS@@A + 0x28
    int32_t rax_1 = UserIsCurrentSessionHostServiceSession()
    int32_t rust_gdi_feature_flag_state
    long rax_4
    if (rax_1 ≠ 0)
        rust_gdi_feature_flag_state = Feature_Rust_GDI_REGION__private_IsEnabled()
        if (rust_gdi_feature_flag_state == 0)
            *data_1c02ad510 = 0
        else
            int64_t rax_3 = Win32AllocPoolZInit(0x10, 'RstG')
            int64_t* rdx_1 = data_1c02ad510
            *rdx_1 = rax_3
            if (rax_3 == 0)
                rax_4 = -0x3fffffe9
            else
                *(rax_3 + 8) = 0
                long rax_5 = AllocateAndLoadBaseRustExports("\SystemRoot\System32\win32kbase_rs.sys", *rdx_1)
                if (rax_5 s< 0)
                    Win32FreePool(*data_1c02ad510)
                rax_4 = rax_5
    if ((rax_1 ≠ 0 && rust_gdi_feature_flag_state == 0) || rax_1 == 0)
        rax_4 = 0
    return rax_4
```

Needs more reversing

- Actually reversing the rgncore::RegionCore, rgncore::scan code
- Code in win32kbase which loads the Rust code
- More recent builds

Rust

```
  scan_count: usize,
  scan_data: Vec<i32>,
  bounds: RECTL,
```

CindA Xiao
@cxiao@infosec.exchange

In the new Rust Windows kernel GDI code, there is a new global allocator registered named `gdi_alloc::Win32Allocator`. It calls `Win32AllocPool` with a fun new pool tag name, "Rust"!

#rust #rustlang #windows #microsoft #reversing #reverseengineering

```
loc::Win32Allocator as core::alloc::global::GlobalAlloc>::alloc::hbe84ca87c2f258d5(int64_t arg1, uint64_t layout_
align u> 0x10) {
anicking::panic::haf36b911569154db(expr: "assertion failed: layout.align()…"  )
n
ALT ol(nBytes: layout_size, tag: 'Rust') __tailcall
```

May 14, 2023 at 4:17:39 AM **(Edited)** · · Elk

1 4 9

CindA Xiao @cxiao@infosec.exchange 3w

For the specific GDI objects, there are still allocations made with the existing GDI-specific pool tags.

It looks like the `rgncore::scan::ScanBuilder<gdi_alloc::TaggedAllocator<_>>` object uses the existing GDI pool tag `Gscn` (i.e. `GDITAG_SCAN_ARRAY`) for vector allocations. (Probably `gdi_alloc::TaggedAllocator<_>` requires specifying a pool tag)

I also see `Gedg` (i.e. `GDITAG_EDGE`) being used in `gdi_rust::region::from_path::GlobalEdgeTable::add_edge`, and `gdi_rust::region::from_path::ActiveEdgeTable::new`, among other places.

#rust #rustlang #windows #microsoft #reversing #reverseengineering

1 1

CindA Xiao @cxiao@infosec.exchange 3w

The Rust Windows kernel GDI code also has symbols for `fallible_vec::FallibleVec<T,A>`, which looks like a non-panicking `Vec` implementation. `try_extend`, `try_extend_from_slice`, `try_splice_in`, and `try_insert` are all implemented.

In fact it looks suspiciously similar to the `rust_fallible_vec` crate, which Microsoft recently open-sourced: github.com/microsoft/rust_fall... 😂

<- Mastodon thread with writeup

infosec.exchange/@cxiao

cxiao.net

github.com/cxiao

Very big thanks again to brentk9 for figuring out how to actually enable the feature flag!