

A blue parallelogram and a light green parallelogram are positioned in the upper-left corner of the slide. The background is a dark navy blue with several lighter blue diagonal stripes running from the bottom-left towards the top-right.

# Let's mess with Windows drivers

Jaanus Kääp  
Clarified Security



# Who, from, what, why?

Who: Jaanus Kääp

From: Clarified Security

What: Researcher & developer

Why: >100 CVE-s, mostly by Adobe & Microsoft

MSRC top list 2016-2020

Like to talk about security



# Drivers research

- Privilege escalation importance
- Main pathways
  - IPC
  - Syscalls
  - Driver communication



# Windows drivers

- Kernel and drivers
- Drivers basic usage
- Driver and device objects
- Types of drivers
- Driver stack



# Communication with drivers

- Connection is made with DEVICE not driver
- Connecting with symbolic link - *registered*
- Connecting with device name - `\\?\GLOBALROOT\Device\%NAME%`
- Basic IO operations - *ReadFile WriteFile*
- Special commands - *DeviceIoControl*
  - Focusing on that



# DeviceIoControl

```
BOOL DeviceIoControl(  
    HANDLE          hDevice,  
    DWORD           dwIoControlCode,  
    LPVOID          lpInBuffer,  
    DWORD           nInBufferSize,  
    LPVOID          lpOutBuffer,  
    DWORD           nOutBufferSize,  
    LPDWORD         lpBytesReturned,  
    LPOVERLAPPED    lpOverlapped  
  
)
```



# DeviceIoControl

```
BOOL DeviceIoControl(  
    HANDLE          hDevice,  
    DWORD           dwIoControlCode,  
    LPVOID          lpInBuffer,  
    DWORD           nInBufferSize,  
    LPVOID          lpOutBuffer,  
    DWORD           nOutBufferSize,  
    LPDWORD         lpBytesReturned,  
    LPOVERLAPPED    lpOverlapped  
)
```



# DeviceIoControl

```
BOOL DeviceIoControl(  
    HANDLE          hDevice,  
    DWORD          dwIoControlCode,  
    LPVOID          lpInBuffer,  
    DWORD           nInBufferSize,  
    LPVOID          lpOutBuffer,  
    DWORD           nOutBufferSize,  
    LPDWORD         lpBytesReturned,  
    LPOVERLAPPED    lpOverlapped  
)
```





# DeviceIoControl

```
BOOL DeviceIoControl(  
    HANDLE          hDevice,  
    DWORD           dwIoControlCode,  
    LPVOID           lpInBuffer,  
    DWORD           nInBufferSize,  
    LPVOID          lpOutBuffer,  
    DWORD           nOutBufferSize,  
    LPDWORD          lpBytesReturned,  
    LPOVERLAPPED    lpOverlapped  
  
)
```



# DeviceIoControl

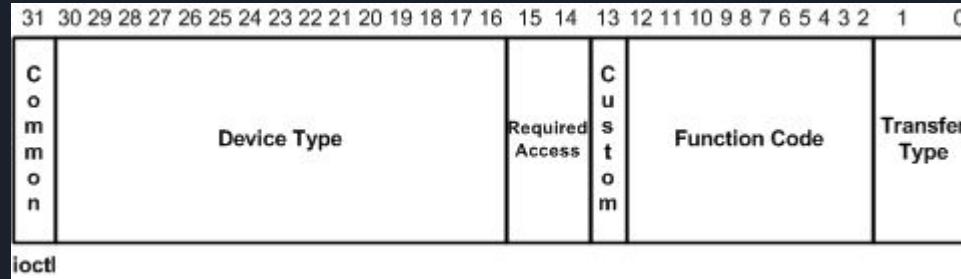
```
BOOL DeviceIoControl(  
    HANDLE          hDevice,  
    DWORD           dwIoControlCode,  
    LPVOID          lpInBuffer,  
    DWORD           nInBufferSize,  
    LPVOID           lpOutBuffer,  
    DWORD           nOutBufferSize,  
    LPDWORD         lpBytesReturned,  
    LPOVERLAPPED    lpOverlapped  
)
```



# DeviceIoControl

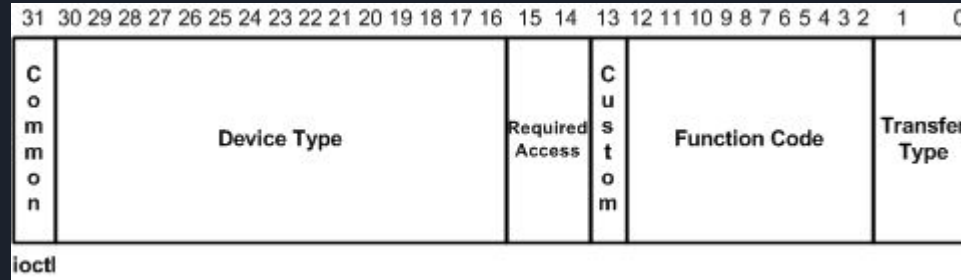
```
BOOL DeviceIoControl(  
    HANDLE          hDevice,  
    DWORD           dwIoControlCode,  
    LPVOID          lpInBuffer,  
    DWORD           nInBufferSize,  
    LPVOID          lpOutBuffer,  
    DWORD           nOutBufferSize,  
    LPDWORD          lpBytesReturned,  
    LPOVERLAPPED    lpOverlapped  
  
    )
```

# DeviceIoControl code



- Device type - as name says (>0x8000 for non MS)
- Requires access - none, read, write or read & write
- Function code - integer specifying the function called
- Transfer type - how data is transferred

# DeviceIoControl code : transfer type



- METHOD\_BUFFERED - in & out data is copied into kernel
- METHOD\_IN\_DIRECT - in buffer is provided as MDL
- METHOD\_OUT\_DIRECT - out buffer is provided as MDL
- METHOD\_NEITHER - userland addresses are provided to driver, no additional protection



# Debugging drivers

- Debugging Windows kernel
- Windbg
- `!drvobj %DRIVER NAME OR ADDRESS% %FLAG%`
  - Flag 0x1 - shows device objects
  - Flag 0x2 - shows dispatch routines
  - Flag 0x4 - shows extra info about device objects
- `!devobj %DEVICE NAME OR ADDRESS%`

# Debugging drivers

3: kd> !drvobj spaceport 1

Driver object (ffff820e8c61ec00) is for:  
[\Driver\spaceport](#)

Driver Extension List: (id , addr)

Device Object list:

[ffff820e8c61f060](#)

3: kd> !devobj ffff820e8c61f060

Device object (ffff820e8c61f060) is for:

Spaceport [\Driver\spaceport](#) DriverObject ffff820e8c61ec00

Current Irp 00000000 RefCount 0 Type 00000004 Flags 00000840

SecurityDescriptor [ffffb204998fe3e0](#) DevExt ffff820e8c61f1b0 DevObjExt ffff820e8c61fcb8

ExtensionFlags (0x00000800) DOE\_DEFAULT\_SD\_PRESENT

Characteristics (0x00000100) FILE\_DEVICE\_SECURE\_OPEN

AttachedTo (Lower) ffff820e8acf0db0 [\Driver\PnpManager](#)

Device queue is not busy.

# Debugging drivers

3: kd> !drvobj spaceport 2  
Driver object (ffff820e8c61ec00) is for:  
\Driver\spaceport

DriverEntry: fffff8072ff1c010 spaceport!GsDriverEntry  
DriverStartIo: 00000000  
DriverUnload: fffff8072ff12a00 spaceport!SpUnload  
AddDevice: 00000000

Dispatch routines:

[00]	IRP_MJ_CREATE	fffff8072fea5300	spaceport!SpSuccess
[01]	IRP_MJ_CREATE_NAMED_PIPE	fffff8072af44b80	nt!IopInvalidDeviceRequest
[02]	IRP_MJ_CLOSE	fffff8072fea5300	spaceport!SpSuccess
[03]	IRP_MJ_READ	fffff8072fea3c40	spaceport!SpDispatch
[04]	IRP_MJ_WRITE	fffff8072fea3c40	spaceport!SpDispatch
[05]	IRP_MJ_QUERY_INFORMATION	fffff8072af44b80	nt!IopInvalidDeviceRequest
[06]	IRP_MJ_SET_INFORMATION	fffff8072af44b80	nt!IopInvalidDeviceRequest
[07]	IRP_MJ_QUERY_EA	fffff8072af44b80	nt!IopInvalidDeviceRequest
[08]	IRP_MJ_SET_EA	fffff8072af44b80	nt!IopInvalidDeviceRequest
[09]	IRP_MJ_FLUSH_BUFFERS	fffff8072fea3c40	spaceport!SpDispatch
[0a]	IRP_MJ_QUERY_VOLUME_INFORMATION	fffff8072af44b80	nt!IopInvalidDeviceRequest
[0b]	IRP_MJ_SET_VOLUME_INFORMATION	fffff8072af44b80	nt!IopInvalidDeviceRequest
[0c]	IRP_MJ_DIRECTORY_CONTROL	fffff8072af44b80	nt!IopInvalidDeviceRequest
[0d]	IRP_MJ_FILE_SYSTEM_CONTROL	fffff8072fea3c40	spaceport!SpDispatch
[0e]	IRP_MJ_DEVICE_CONTROL	fffff8072fea3c40	spaceport!SpDispatch
[0f]	IRP_MJ_INTERNAL_DEVICE_CONTROL	fffff8072fea3c40	spaceport!SpDispatch
[10]	IRP_MJ_SHUTDOWN	fffff8072fea3c40	spaceport!SpDispatch
[11]	IRP_MJ_LOCK_CONTROL	fffff8072af44b80	nt!IopInvalidDeviceRequest
[12]	IRP_MJ_CLEANUP	fffff8072fea5300	spaceport!SpSuccess
[13]	IRP_MJ_CREATE_MAILSLOT	fffff8072af44b80	nt!IopInvalidDeviceRequest
[14]	IRP_MJ_QUERY_SECURITY	fffff8072af44b80	nt!IopInvalidDeviceRequest
[15]	IRP_MJ_SET_SECURITY	fffff8072af44b80	nt!IopInvalidDeviceRequest
[16]	IRP_MJ_POWER	fffff8072fea3c40	spaceport!SpDispatch
[17]	IRP_MJ_SYSTEM_CONTROL	fffff8072fea3c40	spaceport!SpDispatch





# Debugging drivers : handler routines

- NTSTATUS DriverIoControl(PDEVICE\_OBJECT DeviceObject, PIRP Irp)
  - DeviceObject - pointer to device object
  - Irp - I/O Request Packets
- IRP structure is LARGE and filled with unions
- What is important?

# Debugging drivers : IRP elements

!irp %IRP ADDRESS%

```
2: kd> !irp rdx
Irp is active with 2 stacks 2 is current (= 0xffff820e9221cef8)
No Mdl: System buffer=ffff820e902e4000: Thread ffff820e910e2080: Irp stack trace.
    cmd flg cl Device  File      Completion-Context
[N/A(0), N/A(0)]
    0  0 00000000 00000000 00000000-00000000

    Args: 00000000 00000000 00000000 00000000
>[IRP_MJ_DEVICE_CONTROL(e), N/A(0)]
    5  0 ffff820e8c61f060 ffff820e95e5fed0 00000000-00000000
    \Driver\spaceport
    Args: 00001b00 00000028 0xe70008 00000000
```

# Debugging drivers : IRP elements

!irp %IRP ADDRESS%

I/O BUFFER

OUTPUT SIZE

INPUT SIZE

```
2: kd> !irp rdx
Irp is active with 2 stacks 2 is current (= 0xfffff820e9221cef8)
No Mdl: System buffer=ffff820e902e4000: Thread ffff820e910e2080: Irp stack trace.
  cmd flg cl Device  File      Completion-Context
  [N/A(0), N/A(0)]
    0  0 00000000 00000000 00000000-00000000

    Args: 00000000 00000000 00000000 00000000
>[IRP_MJ_DEVICE_CONTROL(e), N/A(0)]
  5  0 ffff820e8c61f060 ffff820e95e5fed0 00000000-00000000
  \Driver\spaceport
    Args: 00001b00 00000028 0xe70008 00000000
```

CODE



# Debugging drivers : IRP elements

- Major function: byte at offset 0x0 from pointer at irp+0xB8
  - `db poi(rdx + 0xB8) L1`
- Control code: dword at offset 0x18 from pointer at irp+0xB8
  - `dd poi(rdx + 0xB8) + 0x18 L1`
- Input size: dword at offset 0x10 from pointer at irp+0xB8
  - `dd poi(rdx + 0xB8) + 0x10 L1`
- Output size: dword at offset 0x8 from pointer at irp+0xB8
  - `dd poi(rdx + 0xB8) + 0x8 L1`
- Input/output buffer: pointer at irp+0x18
  - `db poi(rdx + 0x18)`

# Debugging drivers : IRP elements

```
2: kd> db poi(rdx + 0xB8) L1
ffff820e`9221cef8  0e
2: kd> dd poi(rdx + 0xB8) + 0x18 L1
ffff820e`9221cf10  00e70008
2: kd> dd poi(rdx + 0xB8) + 0x10 L1
ffff820e`9221cf08  00000028
2: kd> dd poi(rdx + 0xB8) + 0x8 L1
ffff820e`9221cf00  00001b00
2: kd> dp rdx+0x18 L1
ffff820e`9221cdf8  ffff820e`902e4000
2: kd> db poi(rdx + 0x18)
ffff820e`902e4000  28 00 00 00 3e 17 8a c2-de c1 eb 11 ab a6 80 6e (...>.....n
ffff820e`902e4010  6f 6e 69 63 00 00 00 00-00 00 00 00 00 00 00 onic.....
ffff820e`902e4020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
ffff820e`902e4030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
ffff820e`902e4040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
ffff820e`902e4050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
ffff820e`902e4060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
ffff820e`902e4070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```



# What to look for

- Memory corruptions
- Data leaks
- Logic flaws



# Memory corruption

- Overflow
- Use after free
- Uninitialized values



# Data leak

- Accidental leak
- Out of bound read
- Uninitialized values





# Logic flaws

- Missing access checks
- Skippable checks
- Etc

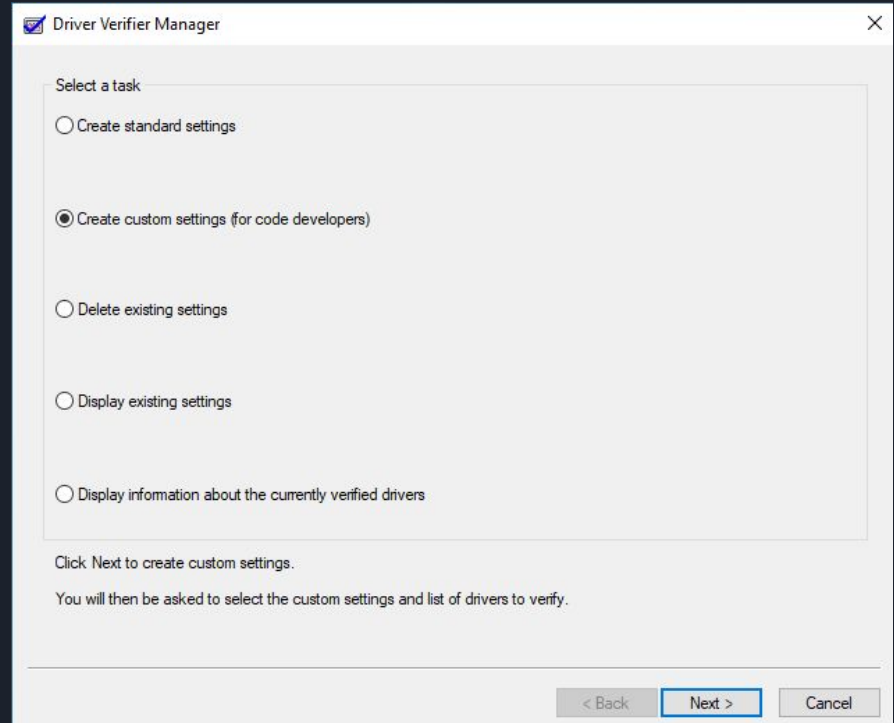


# Finding vulnerabilities

- Manual testing
- Blind dumb fuzzing
- Mutation based fuzzing
- Generation based fuzzing

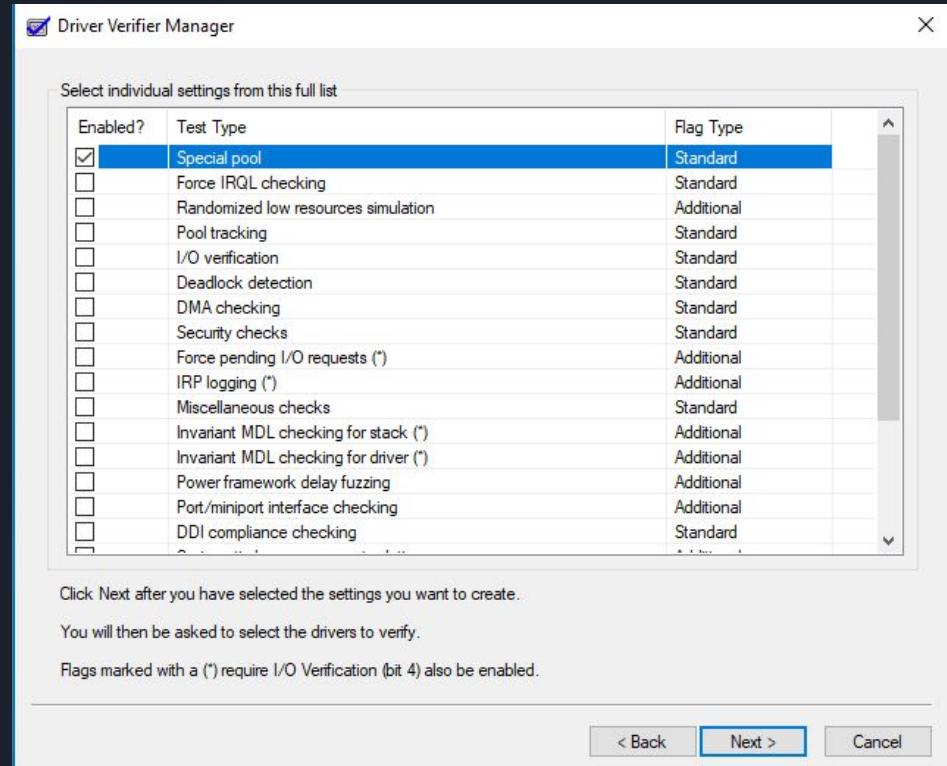
# Single most important prerequisites

- verifier.exe
- Special pool



# Single most important prerequisites

- verifier.exe
- Special pool





# Special pool

- Like page heap or guard pages

PAGE

PAGE

PAGE

PAGE

PAGE



# Special pool

- Like page heap or guard pages





# Special pool

- Like page heap or guard pages



# Special pool

- Like page heap or guard pages
- Nonpageable memory and 2 VM pages
- Can be exhausted
- Additional configuration
  - “Verify start” vs “Verify end”
  - Use by pool tag
  - Use by size







# Finding vulnerabilities

- Manual testing
- Blind dumb fuzzing
- Mutation based fuzzing
- Generation based fuzzing



# Finding vulnerabilities

- **Manual testing**
  - Reverse engineering driver code
  - Combination of static and dynamic analysis
  - Can be combined with fuzzing
  - Often needed anyway to bypass some checks
- Blind dumb fuzzing
- Mutation based fuzzing
- Generation based fuzzing
- Coverage based fuzzing



# Finding vulnerabilities

- Manual testing
- **Blind dumb fuzzing**
  - Stupid easy to setup
  - Small and possibly useless coverage
- Mutation based fuzzing
- Generation based fuzzing
- Coverage based fuzzing



# Finding vulnerabilities

- Manual testing
- Blind dumb fuzzing
- **Mutation based fuzzing**
  - Need to record or create good inputs
  - Could have limits on coverage
- Generation based fuzzing
- Coverage based fuzzing



# Finding vulnerabilities

- Manual testing
- Blind dumb fuzzing
- Mutation based fuzzing
- **Generation based fuzzing**
  - Biggest time investment - RE & dev
  - Covers only area covered by generation logic
  - Can be combined with other methods
- Coverage based fuzzing



# Finding vulnerabilities

- Manual testing
- Blind dumb fuzzing
- Mutation based fuzzing
- Generation based fuzzing
- **Coverage based fuzzing**
  - Most complex to develop
  - Can be combined with other methods
  - Hardware support - Intel PT



# Testing tools

- Lot of existing fuzzers
- RE with Windbg & IDA
- Develop own tools for testing
  - Scripting for quick tests
  - C++ or similar for fast fuzzing
- Bit more generic tool from me :)



# Windows ScrewDriver

- GUI & implementation DLL
- Gives list of drivers and devices
- Can easily make request to drivers
- Can record traffic to drivers
- Can help detect supported codes
- Can help detect expected structure values
- Lot of small functionalities to help RE



# Windows ScrewDriver

The screenshot shows the RYBLIK application window. The title bar reads "RYBLIK". The menu bar includes "File", "Hooking", and "About". The "Main" tab is active. Below the menu bar, there are three checkboxes: "Hide not accessible", "Hide no devices", and "Only RW", all of which are unchecked. To the right of these checkboxes is a "Log file:" text box and a "Start logging" button. The left pane displays a tree view of the Windows Driver File System (DFS). The right pane shows the details for the selected driver, "\Driver\spaceport".

**Major functions**

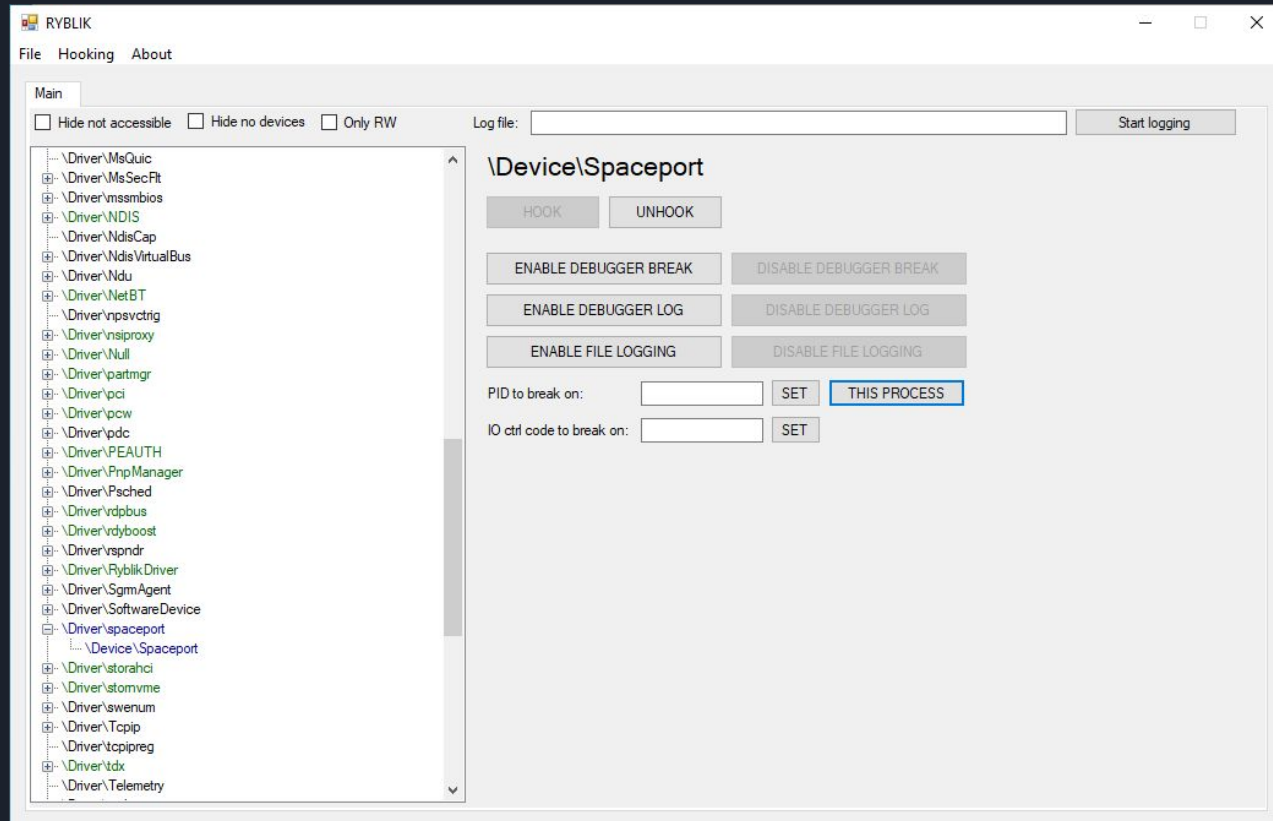
Major function	Pointer
0x00 IRP_MJ_CREATE	0xFFFFF805824A5300
0x01 IRP_MJ_CREATE_NAMED_PIPE	0xFFFFF8057EF44B80
0x02 IRP_MJ_CLOSE	0xFFFFF805824A5300
0x03 IRP_MJ_READ	0xFFFFF805824A3C40
0x04 IRP_MJ_WRITE	0xFFFFF805824A3C40
0x05 IRP_MJ_QUERY_INFORMATION	0xFFFFF8057EF44B80
0x06 IRP_MJ_SET_INFORMATION	0xFFFFF8057EF44B80
0x07 IRP_MJ_QUERY_EA	0xFFFFF8057EF44B80

**Devices**

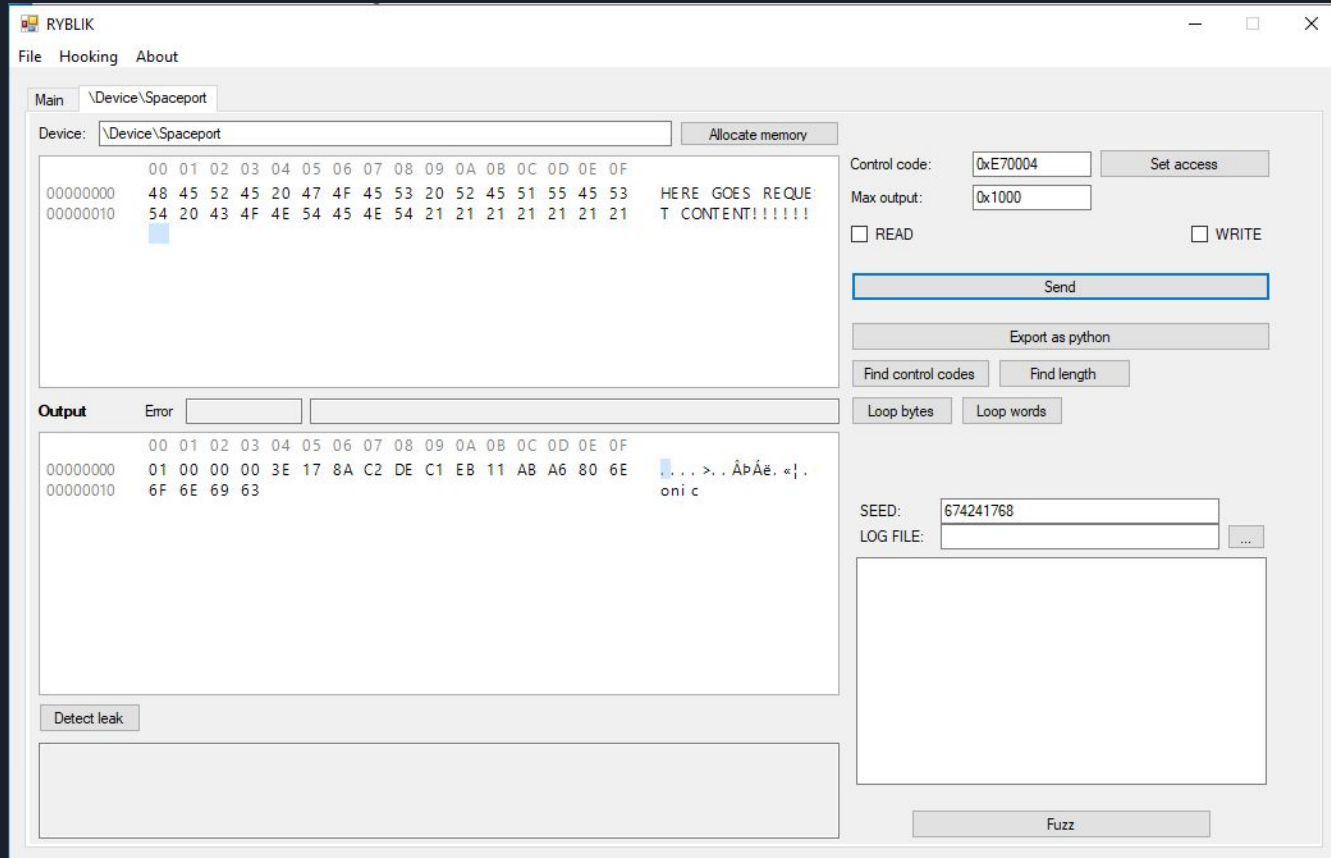
Address	Name	Access
0xFFFFC201CF00F060	\Device\Spaceport	READ & WRITE

At the bottom of the right pane, there are two buttons: "HOOK" and "UNHOOK".

# Windows ScrewDriver



# Windows ScrewDriver



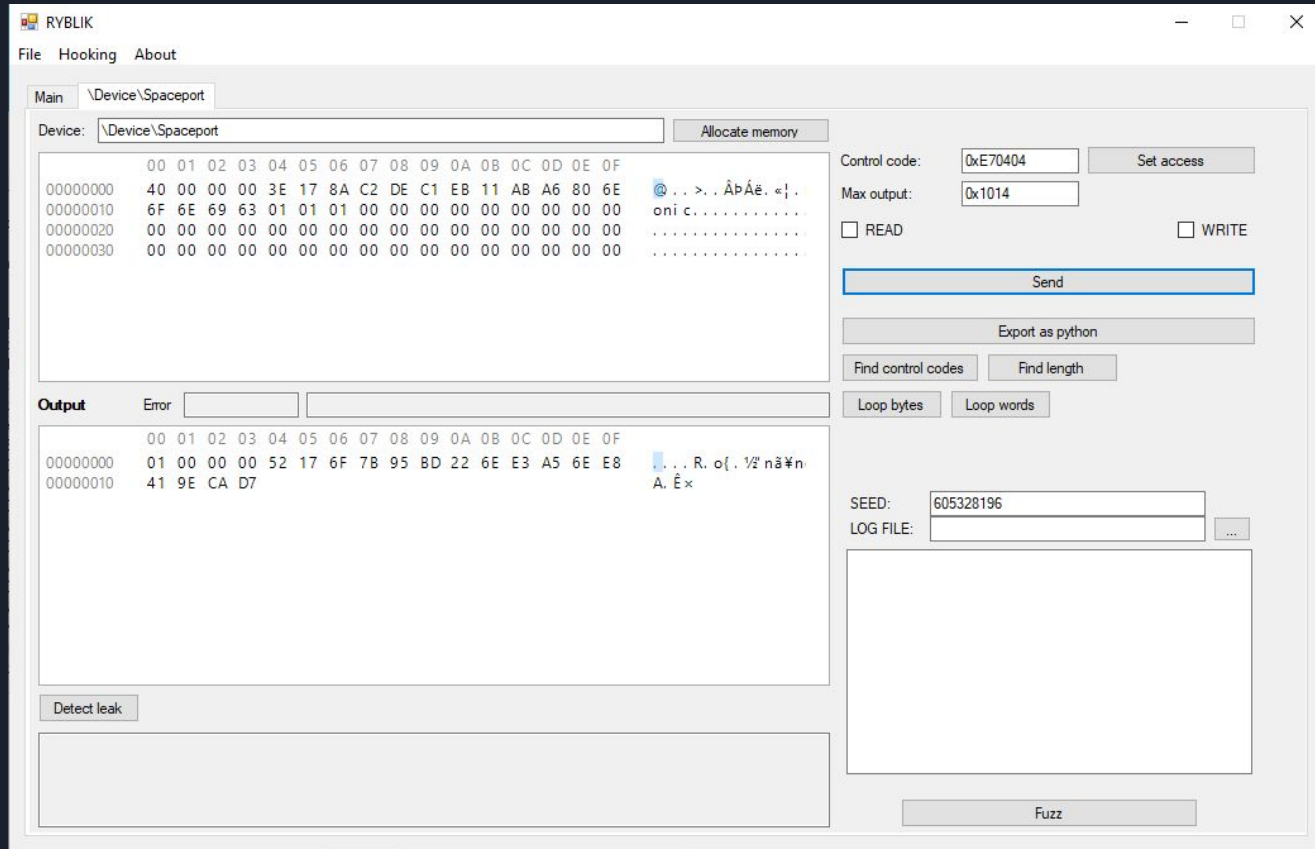
# Windows ScrewDriver

The screenshot displays the Windows ScrewDriver application window, titled "Stored data". The interface is divided into several sections:

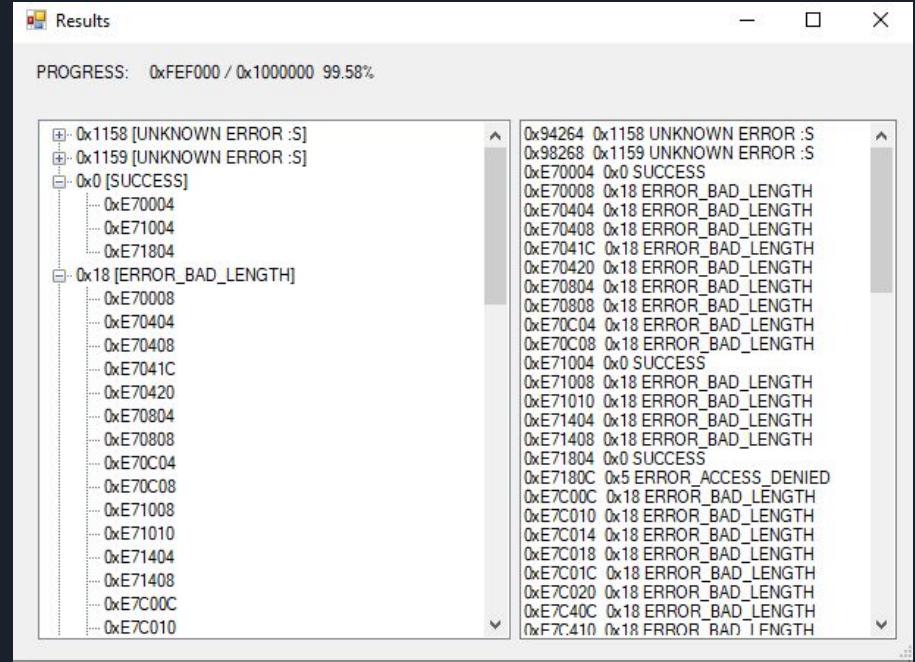
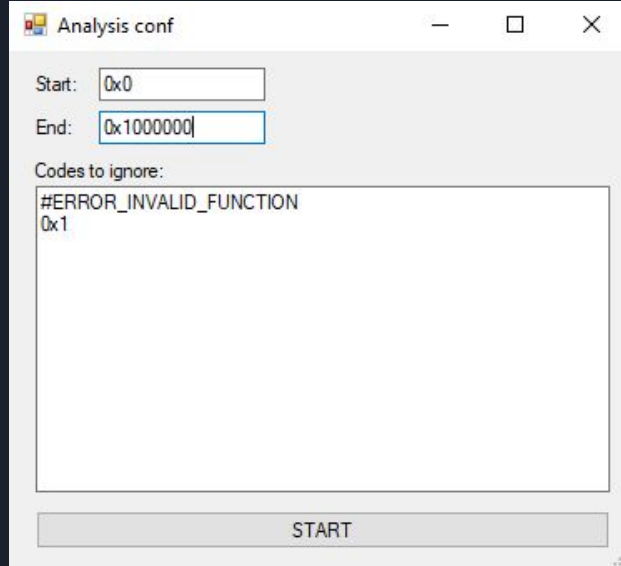
- File Edit**: A menu bar at the top left.
- Tree View**: A hierarchical list of drivers on the left side. The selected driver is `\Driver\spaceport`, which is expanded to show its sub-components: `\Device\Spaceport`, `0xE70004`, `0xE70008`, `0xE70404`, and `0xE70408`. Each sub-component has associated input and output sizes (e.g., `INPUT: 0x0 OUTPUT: 0x1014`).
- Table**: A table on the right side showing details for the selected driver. The table has columns: Index, Driver, Device, Ctrl code, Input size, Output size, and Open. The data is as follows:

Index	Driver	Device	Ctrl code	Input size	Output size	Open
0	\Driver\spaceport	\Device\Spaceport	0xE70004	0x0	0x1014	open
1	\Driver\spaceport	\Device\Spaceport	0xE70008	0x28	0x1B00	open
2	\Driver\spaceport	\Device\Spaceport	0xE70008	0x28	0x1B00	open
3	\Driver\spaceport	\Device\Spaceport	0xE70404	0x40	0x1014	open
4	\Driver\spaceport	\Device\Spaceport	0xE70408	0x28	0x1C28	open
- Delete**: A button located below the table.
- SEED**: A text box containing the value `298984822`.
- LOG FILE**: A text box for specifying a log file.
- Fuzz**: A button for starting the fuzz testing process.
- Store last target info**: A checkbox that is currently checked.

# Windows ScrewDriver



# Windows ScrewDriver





# Windows ScrewDriver

- Helps speed up manual testing
- Can be used to fuzz or create fuzzing base
- Can be used to record valid input for mutational fuzzing or RE
- Will be extended in following years
  - New tools (so it's actually a TOOLSET)



# Windows ScrewDriver

- Helps speed up manual testing
  - Can be used to fuzz or create fuzzing base
  - Can be used to record valid input for mutational fuzzing or RE
  - Will be extended in following years
    - New tools (so it's actually a TOOLSET)
- 
- For BSides Tallinn participants:  
**3 day FREE trial** and after that only 1200EUR/MONTH





# Windows ScrewDriver

- Helps speed up manual testing
  - Can be used to fuzz or create fuzzing base
  - Can be used to record valid input for mutational fuzzing or RE
  - Will be extended in following years
    - New tools (so it's actually a TOOLSET)
- 
- **JUST KIDDING!** It's free and open source under MIT license  
<https://github.com/JaanusKaapPublic/Windows-ScrewDriver>



# What if you find something

- Microsoft Windows Insider Preview Bounty Program
  - Bug has to exist in latest Dev Channel build
  - If device accessible from sandbox, then up to 20 000\$
  - Otherwise up to 5 000\$
  - DON'T believe "Denial of Service \$500" bounty, it's a lie
- ZDI
- Zerodium up to 80 000\$
- Full disclosure to piss of lot of people
- If you find something with my tools
  - Let me know later if possible - it's nice to know
  - Donate something to any reasonable charity

Thank you!

•

