P99 CONF

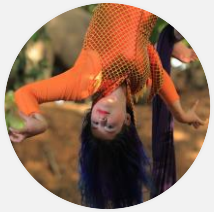
I/O Rings and You - Optimizing I/O on Windows

Yarden Shafir

CrowdStrike

Yarden Shafir

Software Engineer at CrowdStrike



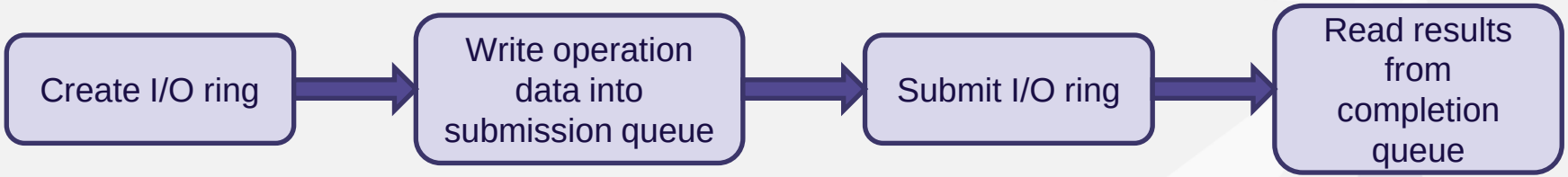
- Software Engineer at CrowdStrike, working on EDR features
- Instructor of Windows Internals classes at Winsider
- Authored articles and tools:
  - CET, Extension Host Hooking, Kernel Exploit Mitigations, PoolViewer
- Circus artist – aerial acrobatics instructor and performer
- Former pastry chef

CROWDSTRIKE

I/O Ring

- A way to asynchronously queue multiple I/O operations
  - Avoids multiple user <-> kernel transitions when performing many I/O operations
- Uses ring buffers for submission and completion queues
- Very similar to io_uring on Linux
  - Implementation on Windows is practically identical
- Documented functions are in KernelBase.dll
    - Header file ioringapi.h in new SDK

Usage Steps


Create I/O ring
Write operation data into submission queue
Submit I/O ring
Read results from completion queue

I/O Ring Creation

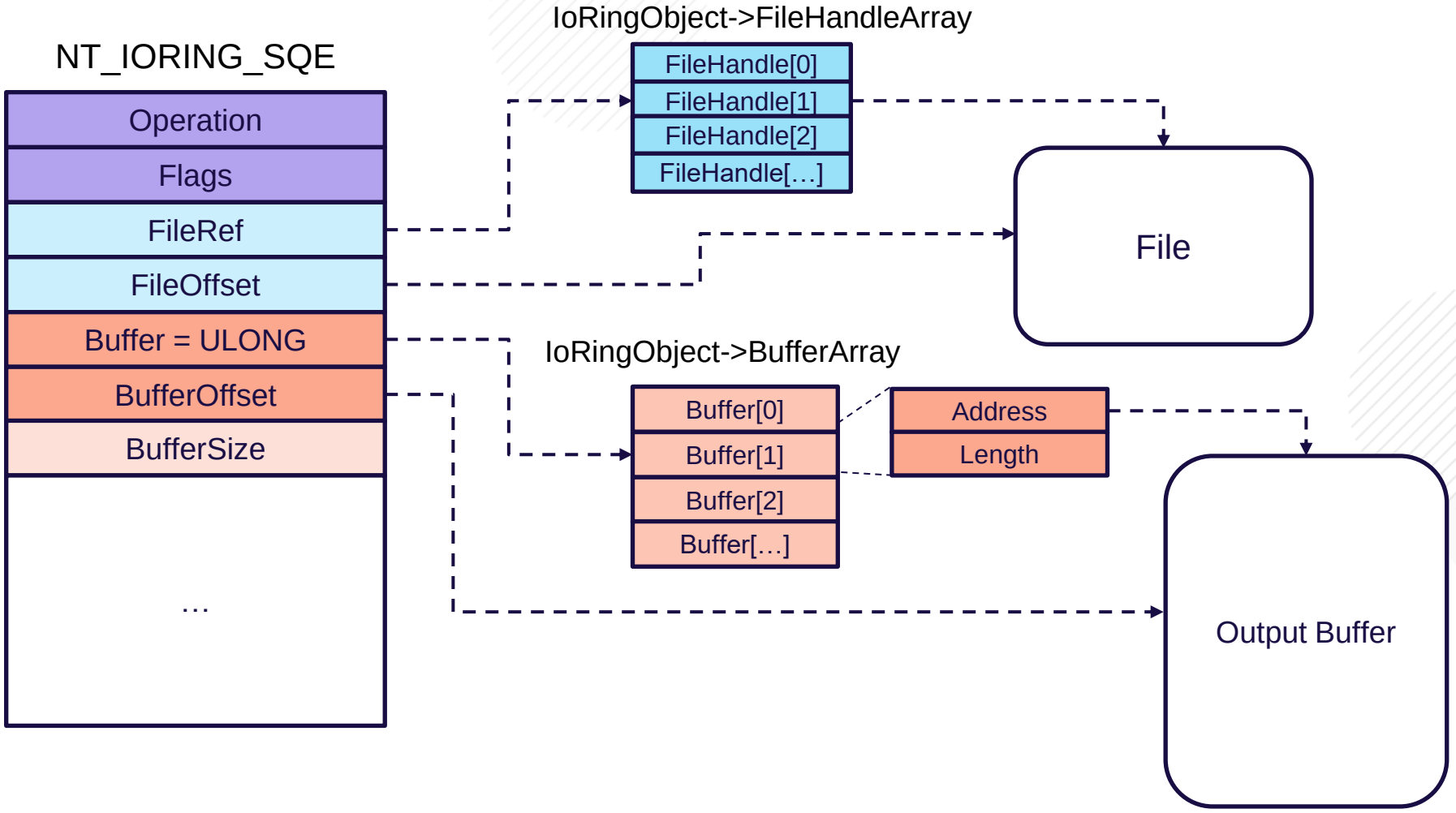
- Created in response to a user request
- IORING_OBJECT is created for the new I/O ring instance
  - Contains all information about this ring and the I/O operations attached to it
- System allocates submission queue and completion queue based on requested sizes
  - Both are in the same section – mapped in both UM and KM
  - Submission queue can have up to 0x10000 entries, completion queue up to 0x20000
- Address and size of new queues are returned to the caller

I/O Ring Operations

- Currently only supported I/O operation is read
- But I/O rings know how to handle several kinds of operation codes:
  - IORING_OP_READ
  - IORING_OP_REGISTER_FILES
    - Receives a pointer to an array of file handles to pre-register
  - IORING_OP_REGISTER_BUFFERS
    - Receives a pointer to an array of buffer + size structures to pre-register
  - IORING_OP_CANCEL

I/O Ring Submissions

- Submission queue starts with a header informing the kernel which entries to process and how many of them
- Header is followed by an array of (undocumented) NT_IORING_SQEs
  - Each contains information about the requested operation: opcode, file handle, buffer, etc
  - File handle can be a handle or an index to a pre-registered array
    - Determined by flag IORING_SQE_PREREGISTERED_FILE
  - Buffer can be a pointer or an index to a pre-registered array
    - Determined by flag IORING_SQE_PREREGISTERED_BUFFER


IoRingObject->FileHandleArray
NT_IORING_SQE
FileHandle[0]
FileHandle[1]
FileHandle[2]
FileHandle[…]
Operation
Flags
FileRef
FileOffset
Buffer = ULONG
BufferOffset
BufferSize
…
File
IoRingObject->BufferArray
Buffer[0]
Buffer[1]
Buffer[2]
Buffer[…]
Address
Length
Output Buffer

Documented Usage – KernelBase.dll

```
IORING_HANDLE_REF requestDataFile(handle);
IORING_BUFFER_REF requestDataBuffer(handle);
IORING_CQE cqe;
HANDLE fileHandle = CreateFile(...);

requestDataFile.Kind = IORING_REF_RAW;
requestDataFile.Handle = fileHandle;
requestDataBuffer.Kind = IORING_REF_RAW;
requestDataBuffer.Buffer = VirtualAlloc(NULL, size, MEM_COMMIT, PAGE_READWRITE);
result = CreateIoRing(IORING_VERSION_1, flags, sqSize, cqSize, &handle);
result = BuildIoRingReadFile(handle, requestDataFile, requestDataBuffer,
                             sizeToRead, 0, NULL, IOSQE_FLAGS_NONE);
result = SubmitIoRing(handle, 1, 0, &submittedEntries);
result = PopIoRingCompletion(handle, &cqe);
```

Performance Analysis

ReadFile	ReadFileEx	I/O Ring Win32 API	I/O Ring NT API
24600140797	23775522267	22698235147	22419537722
20453198839	20413611694	19932095776	19735951191
20623863171	20225322101	20222548679	20185793675
20346912325	20201343017	20017837622	20002724133

- Time in clock ticks to read ~4000 files and sum all their bytes
- On Average, I/O rings are ~2% faster than I/O ports and ~3% faster than synchronous read
  - More accurate testing is needed

The Bottom Line

- I/O rings are cool! And very useful for applications using a lot of I/O
- Similar implementation to Linux provides cross-platform functionality
  - Makes it easier to port code to and from UNIX systems
- Only supports read operations (for now)
  - Other operations will be added in future builds
- Not documented yet -- APIs and internal functionality can change