Building Your Own Debugging Toolbox With ClrMD

Christophe Nasarre @chnasarre
Staff Software Developer

Agenda

- Introduction to ClrMD
- From live process to memory dump
- ClrHeap, addresses and types
- Marshaling data from instance and static fields
- Make it simpler with C# *dynamic*
- Writing a WinDBG extension leveraging ClrMD

Introduction: pick the right tool

Investigation = $$\begin{tikzcd} \textit{Identify} \arrow[r] & \textit{Understand} \arrow[r] & \textit{Verify} \arrow[ll, bend left] \end{tikzcd}$$

Memory issues

Memory profiler (Visual Studio, dotMemory/dotTrace, Perfview)

Performance issues

CPU profiler (Visuals Studio, dotTrace, Perfview)

…and post mortem investigations

Procdump + WinDBG + SOS (not sure you want to go there…)

Introduction: why ClrMD?

ClrMD helps you automate .NET application analysis in C#

Work on running process or memory dump

Sky is your limit!

DEMO

Why ClrMD?

ClrMD Basics

ClrMD = Microsoft.Diagnostics.Runtime Nuget package

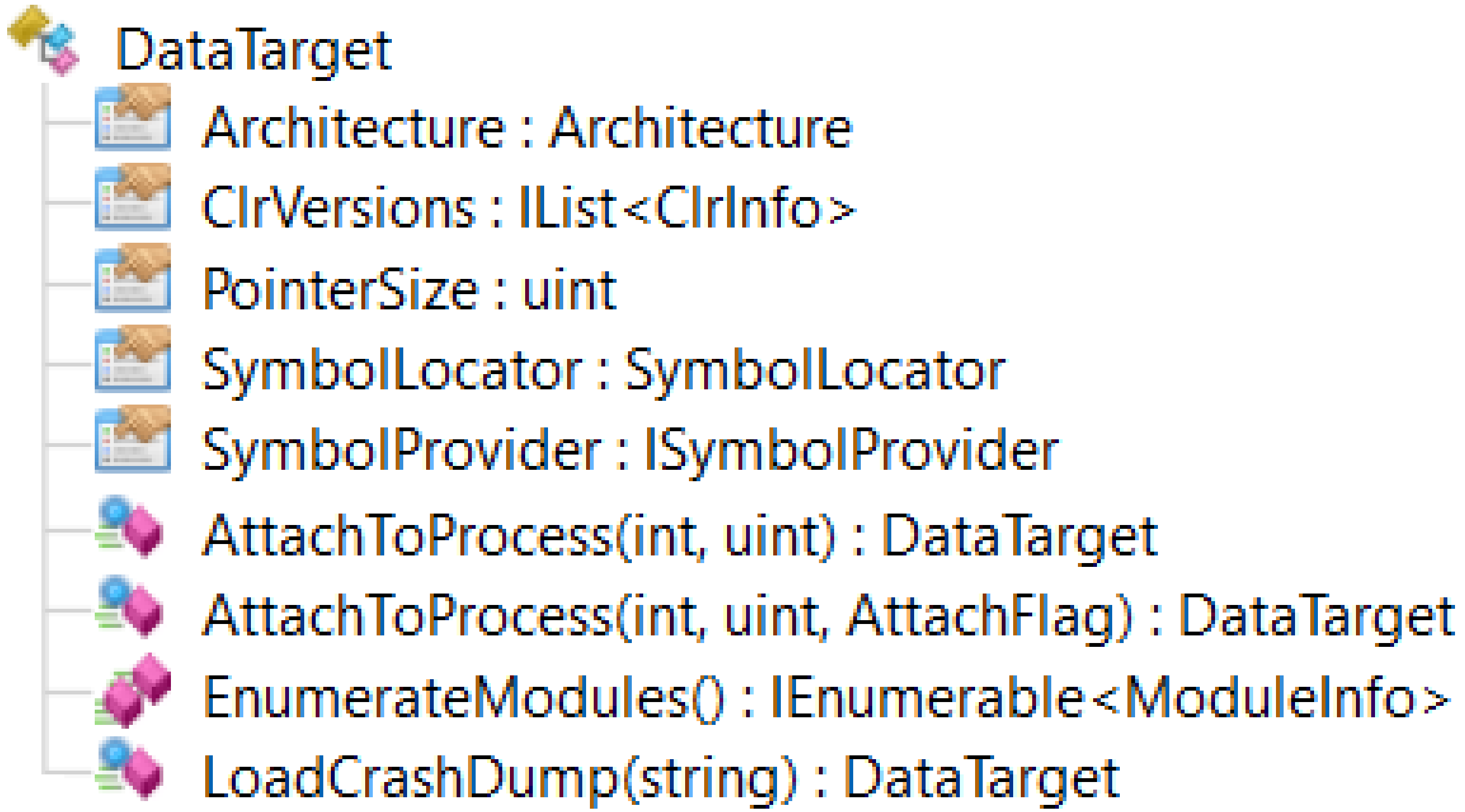
The source code is available on GitHub

Take a look at the samples and the implementation

Agenda

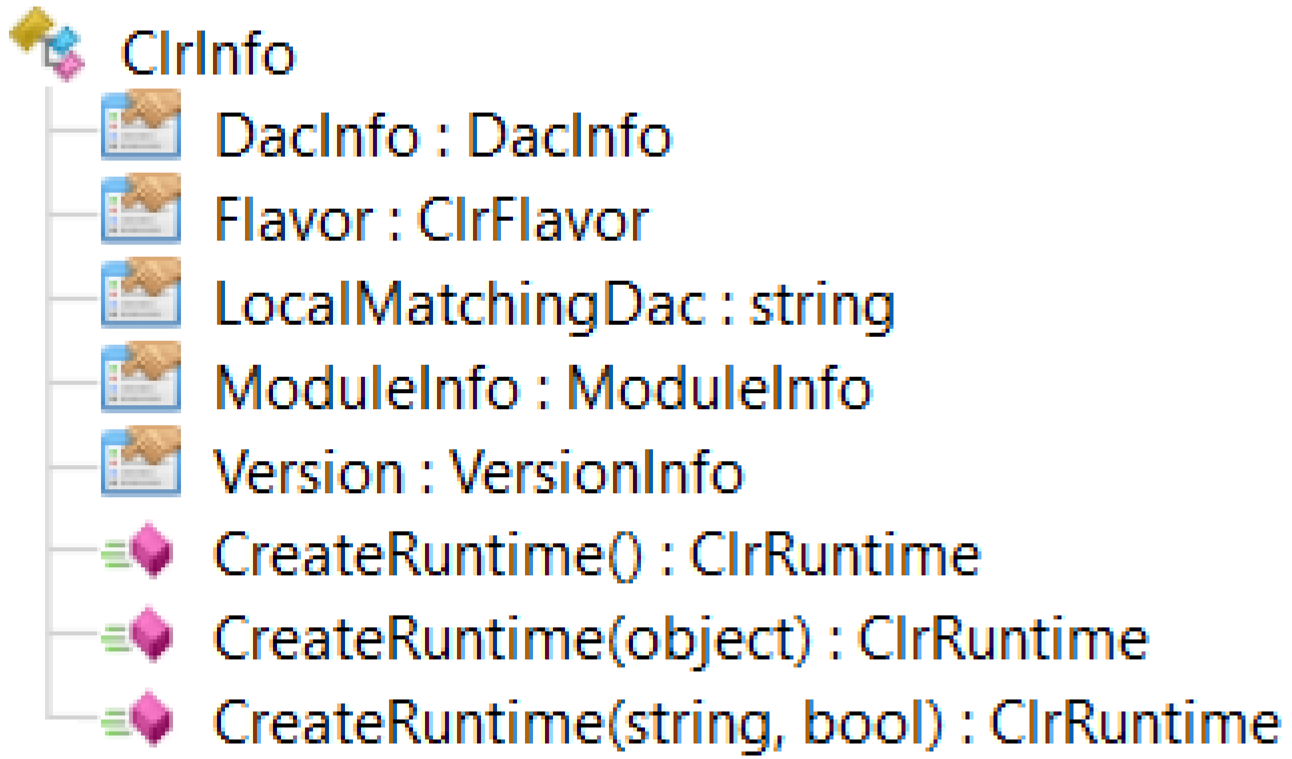
- Introduction to ClrMD
- From live process to memory dump
- ClrHeap, addresses and types
- Marshaling data from instance and static fields
- Make it simpler with C# *dynamic*
- Writing a WinDBG extension leveraging ClrMD

DataTarget to bootstrap them all



```
DataTarget
  Architecture : Architecture
  ClrVersions : IList<ClrInfo>
  PointerSize : uint
  SymbolLocator : SymbolLocator
  SymbolProvider : ISymbolProvider
  AttachToProcess(int, uint) : DataTarget
  AttachToProcess(int, uint, AttachFlag) : DataTarget
  EnumerateModules() : IEnumerable<ModuleInfo>
  LoadCrashDump(string) : DataTarget
```

`ClrInfo` and a little bit of black magic



```
ClrInfo
  DacInfo : DacInfo
  Flavor : ClrFlavor
  LocalMatchingDac : string
  ModuleInfo : ModuleInfo
  Version : VersionInfo
  CreateRuntime() : ClrRuntime
  CreateRuntime(object) : ClrRuntime
  CreateRuntime(string, bool) : ClrRuntime
```

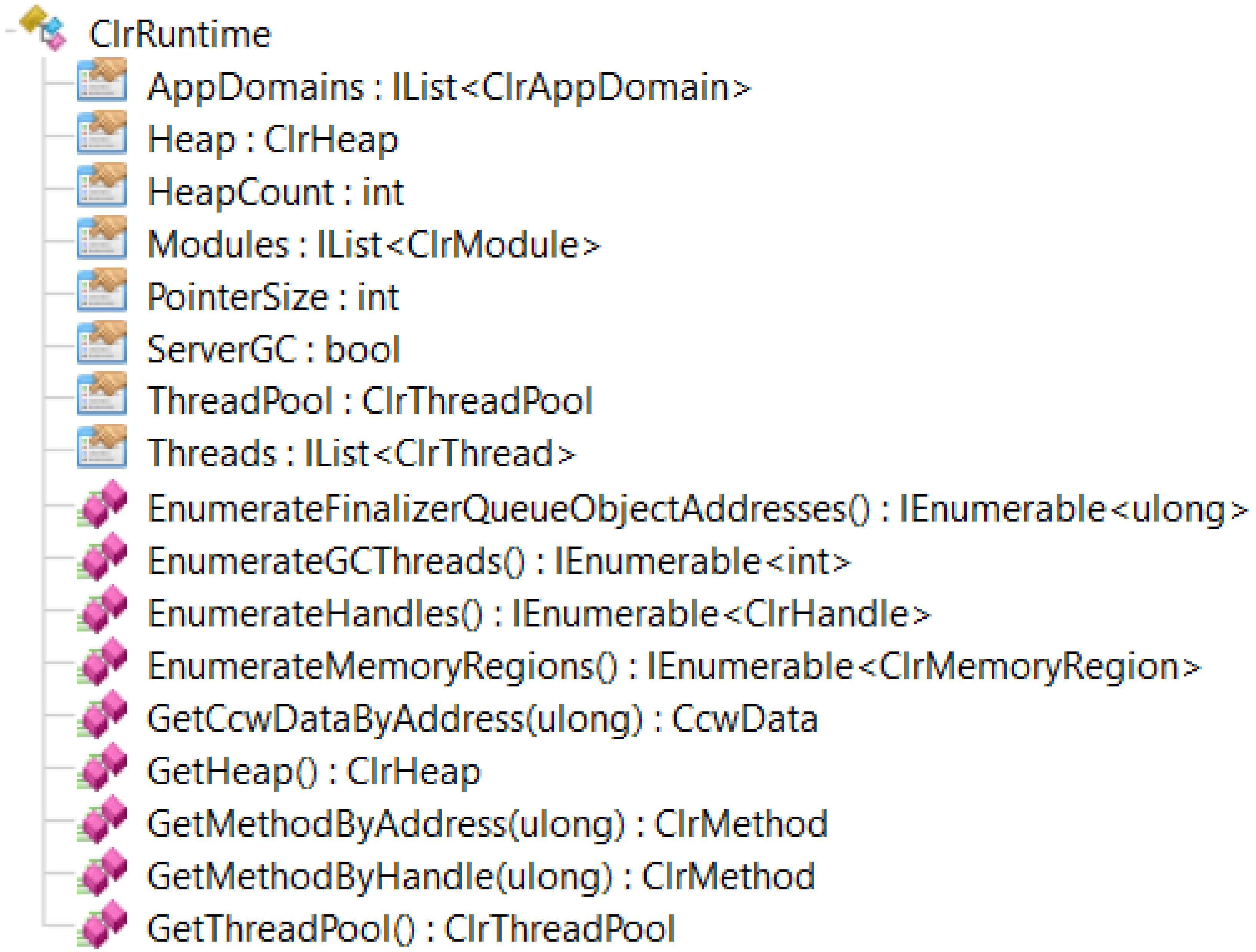
Use **DataTarget.SymbolLocator** to setup symbols/dll locations

srv*c:\symbols*http://msdl.microsoft.com/download/symbols

ClrRuntime

AppDomains
Assemblies (modules)
Threads
Thread Pool
Heap

More advanced
- finalizers
- pinned objects
- methods



```
ClrRuntime
  AppDomains : IList<ClrAppDomain>
  Heap : ClrHeap
  HeapCount : int
  Modules : IList<ClrModule>
  PointerSize : int
  ServerGC : bool
  ThreadPool : ClrThreadPool
  Threads : IList<ClrThread>
  EnumerateFinalizerQueueObjectAddresses() : IEnumerable<ulong>
  EnumerateGCThreads() : IEnumerable<int>
  EnumerateHandles() : IEnumerable<ClrHandle>
  EnumerateMemoryRegions() : IEnumerable<ClrMemoryRegion>
  GetCcwDataByAddress(ulong) : CcwData
  GetHeap() : ClrHeap
  GetMethodByAddress(ulong) : ClrMethod
  GetMethodByHandle(ulong) : ClrMethod
  GetThreadPool() : ClrThreadPool
```

DEMO

Getting started with ClrMD

Agenda

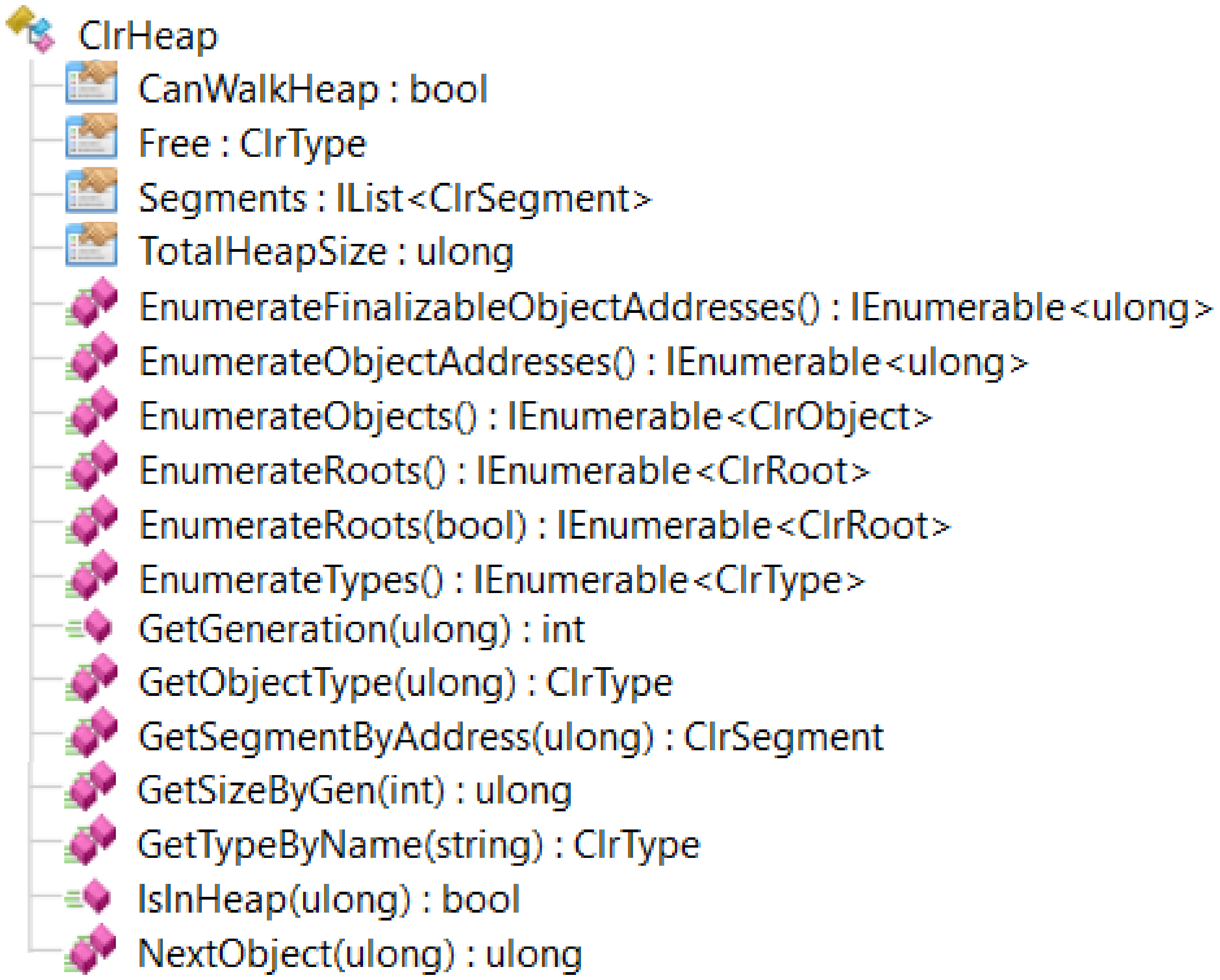
- Introduction to ClrMD
- From live process to memory dump
- ClrHeap, addresses and types
- Marshaling data from instance and static fields
- Make it simpler with C# *dynamic*
- Writing a WinDBG extension leveraging ClrMD

ClrHeap

CanWalkHeap!

address != object

Low level details
- segments
- finalizables
- roots



ClrHeap
- CanWalkHeap : bool
- Free : ClrType
- Segments : IList<ClrSegment>
- TotalHeapSize : ulong
- EnumerateFinalizableObjectAddresses() : IEnumerable<ulong>
- EnumerateObjectAddresses() : IEnumerable<ulong>
- EnumerateObjects() : IEnumerable<ClrObject>
- EnumerateRoots() : IEnumerable<ClrRoot>
- EnumerateRoots(bool) : IEnumerable<ClrRoot>
- EnumerateTypes() : IEnumerable<ClrType>
- GetGeneration(ulong) : int
- GetObjectType(ulong) : ClrType
- GetSegmentByAddress(ulong) : ClrSegment
- GetSizeByGen(int) : ulong
- GetTypeByName(string) : ClrType
- IsInHeap(ulong) : bool
- NextObject(ulong) : ulong

How to browse all objects in the heap

```
foreach (ulong address in heap.EnumerateObjectAddresses())
{
    try
    {
        var objType = heap.GetObjectType(address);
        if (objType == null)
            continue;

        var obj = objType.GetValue(address);

        ...

    }
    catch (Exception x)
    {
        WriteLine(x);
        // some InvalidOperationException might occur sometimes
    }
}
```

DEMO

Count duplicated strings

Agenda

- Introduction to ClrMD
- From live process to memory dump
- ClrHeap, addresses and types
- Marshaling data from instance and static fields
- Make it simpler with C# *dynamic*
- Writing a WinDBG extension leveraging ClrMD

Problem of class instance marshalling

All addresses are meaningless in the current process

`ClrType.GetValue()` automatically marshals basic types

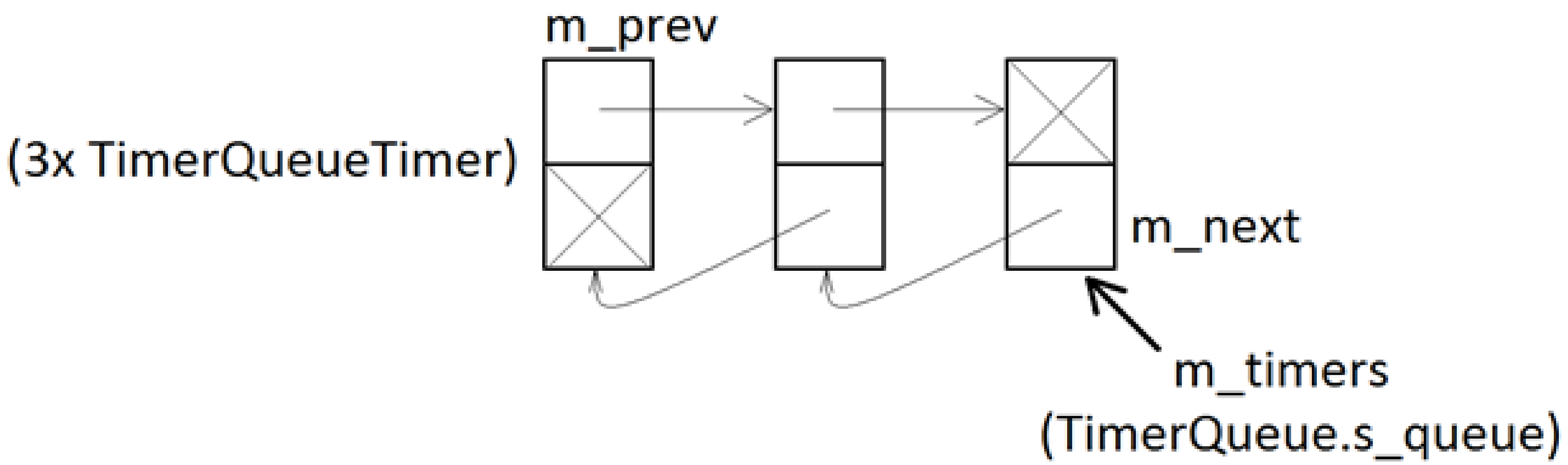
- numbers
- Bool
- String

All reference type instances must be marshalled by hand
→ field by field!

Implementation details of Timer

A **`Timer`** stores its details in a **`TimerQueueTimer`**

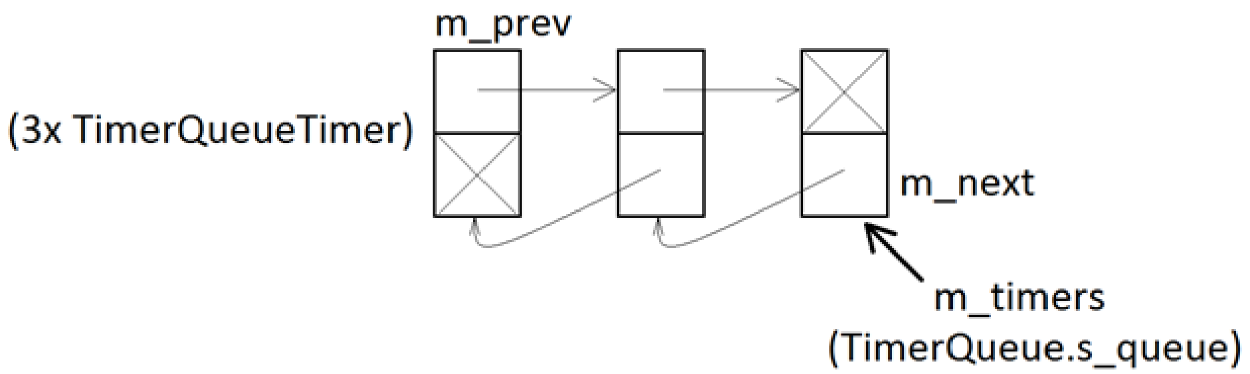
A **static `_queue`** field of **`TimerQueue`** points to the list head



criteo

How to list all timers?

1. Get the **`ClrType`** corresponding to **`TimerQueue`**
2. Reaching the static **`s_queue`** field description
3. Reading the static **`s_queue`** field value to get the list head
4. Reading an instance field to get the next **`TimerQueueTimer`**
5. Decyphering a callback method name and target



How to access a class static field? (1/2)

Access directly to a specific **ClrType**

- Look for the defining module
- Call **ClrModule.GetTypeByName** with the name

```
foreach (ClrModule module in runtime.Modules)
   if (module.AssemblyName.Contains("mscorlib.dll"))
      return module.GetTypeByName("System.Threading.TimerQueue");
```

How to access a class static field? (2/2)

Access a static field via **ClrType.GetStaticFieldByName**

Each App Domain has a different value for all statics

- List all App Domain
- Check if the static has a value or not

```
ClrStaticField staticField =
   timerQueueType.GetStaticFieldByName("s_queue");
foreach (ClrAppDomain domain in runtime.AppDomains)
{
    ulong? timerQueue = (ulong?)staticField.GetValue(domain);
    if (!timerQueue.HasValue || timerQueue.Value == 0)
        continue;
```

How to get instance field value?

Get the **`ClrInstanceField`** from **`ClrType`**

- Get the type from the instance address

Call **`ClrInstanceField.GetValue()`** with the instance address

```
var type = heap.GetObjectType(address);
ClrInstanceField field = type.GetFieldByName(fieldName);
return field?.GetValue(address);
```

How to decipher a delegate?

Difference between an instance and a static method

- Look for the value of **`_target`** field

The callback is stored in the **`_methodPtr`** field

- Use **`ClrRuntime.GetMethodByAddress`** to get a **`ClrMethod`**

```
var methodPtr = GetFieldValue(heap, timerCallbackRef, "_methodPtr");
ClrMethod method = clr.GetMethodByAddress((ulong)(long)methodPtr);
var thisPtr = GetFieldValue(heap, timerCallbackRef, "_target");
if ((thisPtr != null) && ((ulong)thisPtr) != 0)
{
    ...
```

DEMO

get field value and method

Agenda

- Introduction to ClrMD
- From live process to memory dump
- ClrHeap, addresses and types
- Marshaling data from instance and static fields
- Make it simpler with C# *dynamic*
- Writing a WinDBG extension leveraging ClrMD

ClrMD-based code can be verbose

ClrMD is powerful but syntax can be tedious to use:

```csharp
var type = heap.GetObjectType(address);
ClrInstanceField field = type.GetFieldByName("value");
return field?.GetValue(address);
```

What if we could use an easier syntax?

```csharp
return heap.GetProxy(address).value;
```

Problems with ClrMD

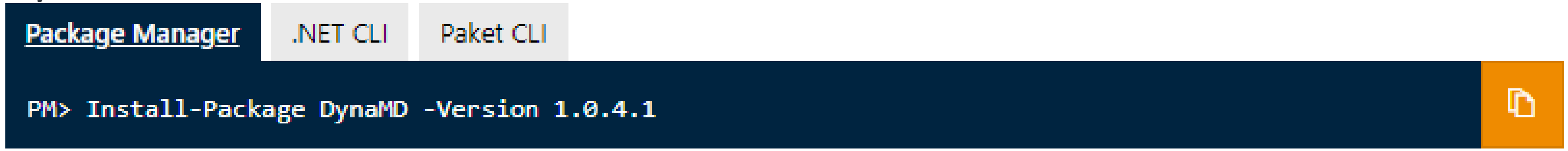
- Verbose syntax to get any value
  - Need a `ClrType`
  - Marshal everything explicitly
- Lack of enumeration/array iterator
  - Where are my `for`/`foreach`?
- Missing boilerplate helpers
  - How to get all instances of a type?

Late-binding in C#

`dynamic` keyword enables the usage of late-binding in C#

Implement `DynamicObject` and override `TryGetMember`, `TryInvokeMember`, `TryConvert` and `TryGetIndex` as needed

DynaMD does all that, and a bit more



```
PM> Install-Package DynaMD -Version 1.0.4.1
```

DynaMD or accessing objects in C#

Wrap remote objects with **`DynamicProxy`**

```
var obj = heap.GetProxy(address);
```

Access object fields a-la C#

```
var buckets = obj.m_tables.m_buckets
```

Allow **`foreach`** on **`IEnumerable`** and **`for`** on arrays

Easy to wrap

```
var queues = heap.GetProxies(concurrentQueueTypeName);
```

DEMO

Look at DynaMD usage

Agenda

- Introduction to ClrMD
- From live process to memory dump
- ClrHeap, addresses and types
- Marshaling data from instance and static fields
- Make it simpler with C# *dynamic*
- Writing a WinDBG extension leveraging ClrMD

WinDBG Extension 101

Extension = .dll exporting commands as native functions

- Case sensitive
- Provide long and short command names
- Even the !help command

Use UnmanagedExports nuget to export managed methods

- Decorate your static methods with DllExport attribute

```
MyCmd(IntPtr client, [MarshalAs(UnmanagedType.LPStr)] string args)
```

Copy your extension + dependencies into winext subfolder

Bind ClrMD with WinDBG extension

Add Common.cs from Github WinDbgExt sample

- Resolve dependency to ClrMD thanks to **`AppDomain.AssemblyResolve`**
- Expose **`DebugExtensionInitialize`** function for versioning
- Bind the **`Console.Write/WriteLine`** output to WinDBG output

Extend the DebuggerExtensions partial class with your commands

- Just call **`InitApi()`** with the received **`IDebugClient`**

DEMO

Show ClrMD GitHub common.cs

Resources

Criteo blog series and source code

- http://labs.criteo.com/2017/12/clrmd-part-9-deciphering-tasks-thread-pool-items/
- https://github.com/chrisnas/DebuggingExtensions

ClrMD on github for source code and samples

https://github.com/Microsoft/clrmd

DynaMD on github

https://github.com/kevingosse/DynaMD