WinDbg Data Model

Time to put the @ back in the bag

Debugger Data Model

- Accessible through DX command
- Create native debugger objects (NatVis)
  - There are built-in objects but user can also define their own with XML
- Can interact and query objects with LINQ
  - Query language built on top of database languages such as SQL
  - Can use Where, Select, OrderBy, etc
- Allows doing complicated actions without MASM
  - So not like this:
    .foreach (place {.shell -ci "!object \\" sed 1,8d | sed s/" .. "//g | sed s/" .*"//g}) {r $t0 = place; r $t1 = $t0-28; dt nt!_OBJECT_HEADER_NAME_INFO Name @$t1}

Built-in Registers:

- @$curthread
- @$curprocess
- @$cursession
- @$curstack
- @$curframe
- Examples:
  - dx (@$cursession.Processes.Where(p => p.Name == "explorer.exe").First()).KernelObject.SignatureLevel
  - dx -r2 @$cursession.Processes.Select(p => p.KernelObject.SignatureLevel)

Anonymous types

- Allow dynamically defining unnamed types for single use without using XML
- Example:
  - dx -r2 @$cursession.Processes.Select(p => new {Name = p.Name, SignatureLevel = p.KernelObject.SignatureLevel}).OrderBy(p => p.SignatureLevel)

Breakpoints

- Conditional breakpoints – bp /w “dx command” <address>
- Example:
  - bp /w "@$curthread.KernelObject.ClientSecurity.ImpersonationData != 0" nt!NtOpenFile
- As always, can add actions to be done when breakpoint is hit
- Example:
  - bp /w "@$curthread.KernelObject.ClientSecurity.ImpersonationData != 0" nt!NtOpenFile "dx @$curthread.KernelObject.ClientSecurity; g"