WinDbg Superpowers For .NET Developers

Sasha Goldshtein @goldshtn
CTO, Sela Group github.com/goldshtn

Agenda

- Miscellaneous tips for making WinDbg friendlier
- Mastering the superpower of scripts and breakpoints
- Useful extensions to avoid tedious work
- Startup and remote debugging sessions
- Visual Studio can’t do 90% of what we’ll see today (or more)

Is Visual Studio A Powerful Debugger?

- **It's a toy for people who like clicking things with the mouse**
- No macros
- No scripts
- No extensions*
- Almost completely reliant on source code being present

Debugging Times

06 DEC 2016

Visual Studio Named Best Debugger

By MICROSOFT CORP

Visual Studio wins best debugger title once again, in a heated competition with the Eclipse debugger, Xamarin Studio, and printf-statements. This notorious victory has been celebrated in Redmond and the entire Pacific Northwest. More on page 17.

Visual Studio

Reuters

International Moose Count Underway

By BOB O'BOBSTON

to make gains. The largest percentagege increase in moose will likely come from China'', says McRobson, The Chinese government has invested heavily in moose infrastructure over the past decade, and their committment to macrofauna is beginning to pay dividends''. Since 2004 China has expanded moose pasture from 1.5% of arable land to nearly 3.648% and moose numbers are expected to rise to 60,000 making China a net moose exporter for the first time. This is good news for neighbouring Mongolia, a barren moose-wasteland whose inhabitents nonetheless have an insatiable desire for the creatures. The increase in Beijing-Ulanbataar trade is anticipated to relieve pressure on the relatively strained Russian suppliers, but increase Mongolia's imbalance of trade with its larger neighbour.

Historically the only competitor to China in the far eastern moose markets has been Singapore but the tiny island nation is set to report a net loss, expecting a decrease of more than five percent on last year's 50,000 moose counted. The head of Singapore's Agency for Agriculture, Jing-Feng Lau, explained to an incredulous Singaporean parliament yesterday that bad weather had contributed

ity controls are holding back the development of the eastern european populations compared to last year when they contributed significantly to europe's strong growth figures. Norway, which is not an EU member but has observer status, strengthed in numbers relative to the Euro area with numbers of Norweigian moose, known locally as elk'' expected to rise for the tenth consecutive year, particularly thanks to a strong showing in the last quarter.

As moose season reaches its close, researchers world wide are turning to science in an attempt to boost next year's figures. NASA stunned the scientific community today with the announcment of their discovery that the moon is significantly smaller than previously believed. This conclusion, which is the conclusion of a ten-year collaborative project, will have profound implications for the moose community as the gravitational field is now known to be of the right strength to support moose in orbit.

According to John Johnson, head of the NASA Moon Sizing Experiment the first delivery of moose into low moon orbit could be achieved as early as the third quarter of next year. The technology to nurture moose in

Originally posted on Reddit

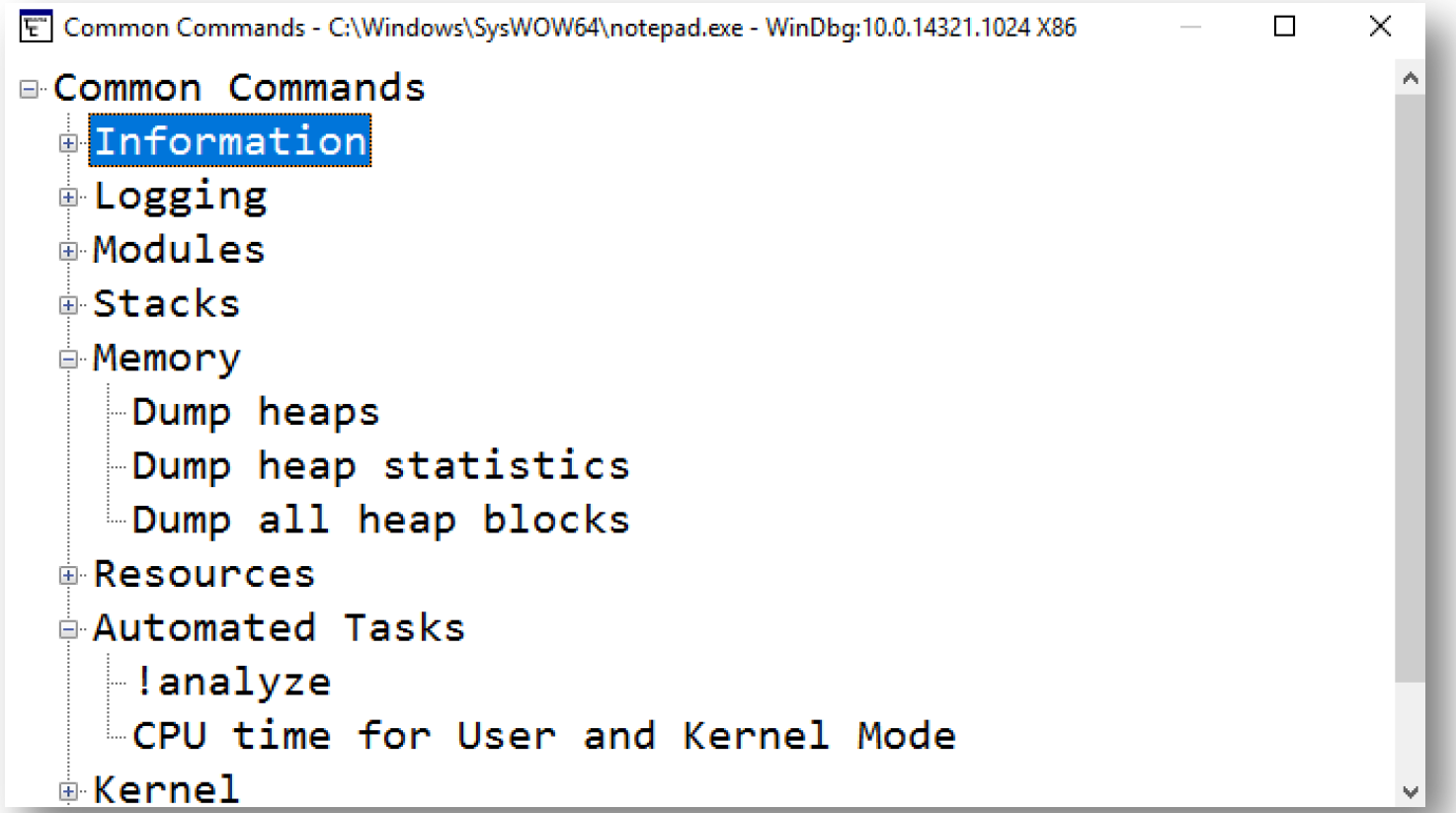
WhoCaresAboutThat 2015


MAKE WINDBG
GREAT AGAIN

Why Is WinDbg So Scary?

- Because I can't remember all the obscure ugly commands
- `.cmdtree` to the rescue!

```
0:000> .cmdtree cmdtree.txt
```



Why Is WinDbg So Scary?

- Because commands are not discoverable, and require copy-pasting lots of hex numbers
- `.prefer_dml 1` (and a recent version of WinDbg) to the rescue!

```
0:018> !DumpClass /d 04dd3a94
Class Name:      MemoryExhaustingService.Product
mdToken:         02000002
File:            V:\courses\Debugging\Exercises\23_MemoryLeak\Binaries\MemoryExhaustingService.exe
Parent Class:    72333d14
Module:          00f13fdc
Method Table:    04e05018
Vtable Slots:    4
Total Method Slots:  5
Class Attributes:    100001
Transparency:        Critical
NumInstanceFields:   3
NumStaticFields:     0
      MT    Field   Offset                 Type VT     Attr    Value Name
727ef7a4  4000001        4        System.String  0 instance           <Name>k__BackingField
727f1638  4000002        c         System.Int32  1 instance           <Stock>k__BackingField
727f4448  4000003        8       System.Byte[]  0 instance           <Image>k__BackingField
```

Why Is WinDbg So Scary?

- Because it takes so many commands to get even basic information out of a dump or a live process
- `windbg -c` to the rescue!

```
C:\> cdb -z mydump.dmp -c ".logopen analysis.txt; !analyze -v; .logclose"
…
C:\> findstr "PROCESS_NAME FAILURE_BUCKET_ID OSEDITION" analysis.txt
PROCESS_NAME:  FileExplorer.exe
FAILURE_BUCKET_ID:
CLR_EXCEPTION_System.NullReferenceException_80004003_FileExplorer.exe!
FileExplorer.MainForm+__c__DisplayClass1._treeView1_AfterSelect_b__0
OSEDITION:  Windows 10 WinNt SingleUserTS
```

Why Is WinDbg So Scary?

- Because it takes so many commands to get even basic information out of a dump or a live process
- `windbg -c` to the rescue!

```
C:\> cdb -pn service.exe -c
     ".loadby sos clr; !dumpheap -stat -min 10000; qd"
…
Statistics:
      MT    Count    TotalSize Class Name
05755068        1        65548 MemoryExhaustingService.Product[]
01600b00       11      3924762      Free
727f4448        5     34013244 System.Byte[]
Total 17 objects
```

Scripting

- A debugger without scripts is a program without loops
- Don’t be discouraged by WinDbg’s horrible scripting language!

```
0:000> r $t0 = 0
0:000> bp ntdll!NtAllocateVirtualMemory "r $t0 = @$t0 + dwo(@rdx); gc"
0:000> g
0:000> .printf "allocated total %d bytes of virtual memory\n", @$t0
allocated total 232191120 bytes of virtual memory
0:000> .for (r $t0 = 0; @$t0 < 0n10; r $t0 = @$t0 + 1)
       { .printf "%x ", @@(arr[@$t0]) }
10 20 30 40 50 60 70 80 90 100
```

Why Am I Creating These Files?

- My app is creating unnecessary files that I can't get rid of

Just Place A Breakpoint™

```
0:000> bp kernelbase!CreateFileW
       ".printf \"Opening file %mu\", dwo(@esp+4); .echo ---; k 3; gc"
0:000> bp kernelbase!CreateFileA
       ".printf \"Opening file %ma\", dwo(@esp+4); .echo ---; k 3; gc"
0:000> g
Opening file V:\logs\SynA86C.tmp---
ChildEBP RetAddr
0918f93c 002b1a5e KERNELBASE!CreateFileW
0918fd7c 002b1725 BatteryMeter!CPUInformation::CPUInformation+0x5e
0918fdbc 750962c4 BatteryMeter!TemperatureAndBatteryUpdaterThread+0x95
```

Why Can't I Open These Files?

- My app complains about some missing files, but I can't figure out where in my code I'm trying to open them

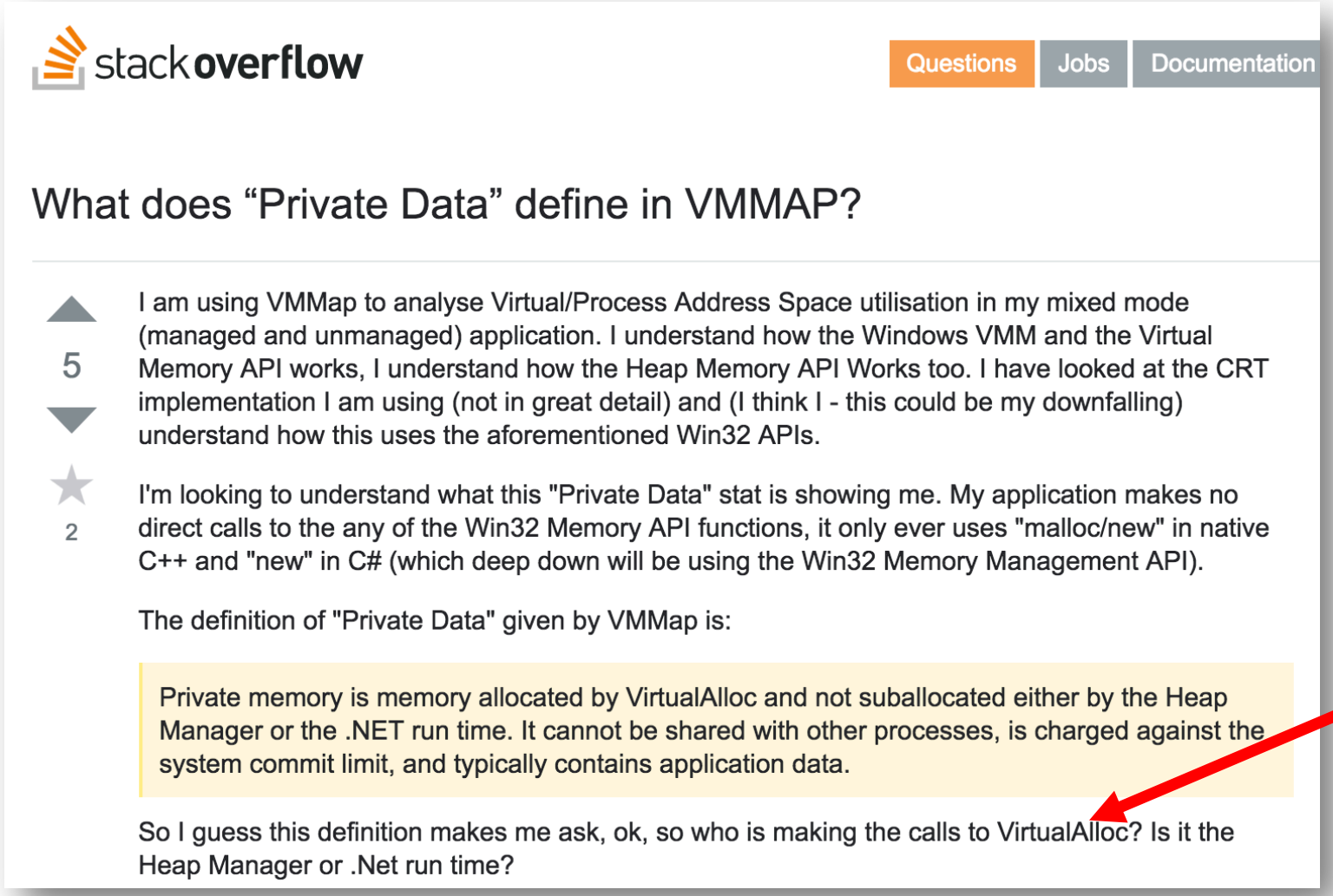
Just Place A Breakpoint™

```
0:000> bp kernelbase!CreateFileW+0x61
       "gu; .if (@eax == 0) {
            .printf \"failed to open file=%mu\", dwo(@esp+4); .echo ---; k3
        } .else { gc }"

0:000> g

Failed to open file=V:\logs\SynA86C.tmp---
ChildEBP RetAddr
0918f93c 002b1a5e KERNELBASE!CreateFileW
0918fd7c 002b1725 BatteryMeter!CPUInformation::CPUInformation+0x5e
0918fdbc 750962c4 BatteryMeter!TemperatureAndBatteryUpdaterThread+0x95
```

Who Is Calling This Function?



Who Is Calling This Function?

- You know the drill…

```
0:000> bp kernelbase!VirtualAlloc
       ".printf \"allocating %d bytes of virtual memory\", dwo(@esp+8);
        .echo; k 5; !clrstack"

0:000> g
allocating 65536 bytes of virtual memory
ChildEBP RetAddr
07bce7a0 7371d339 KERNELBASE!VirtualAlloc
07bce7cc 7371d364 clr!EEVirtualAlloc+0xa0
07bce7dc 73887a55 clr!CExecutionEngine::ClrVirtualAlloc+0x14
07bce80c 73887ad0 clr!WKS::virtual_alloc_commit_for_heap+0x74
07bce820 73887b2d clr!WKS::gc_heap::grow_heap_segment+0x7f
Child SP       IP Call Site
07bce9e8 74643460 [HelperMethodFrame: 07bce9e8]
07bcea74 691e53ba System.Xml.XmlDictionaryReader.ReadContentAsBase64()
07bcea94 691e5335 System.Xml.XmlBaseReader.ReadContentAsBase64()
07bceaa8 691e5252 System.Xml.XmlDictionaryReader.ReadElementContentAsBase64()
…
```

Where Is That Interesting Object?

- Which `Order` object in this 10000-item list has an `Address` property that contains the character å which breaks my encoding?
- Hold on:

```
0:006> !name2ee OrderService!OrderService.Order
EEClass:     01591830
Name:        OrderService.Order

0:006> !dumpclass 01591830
MT       Offset    Type            VT  Attr     Name
727f1638 c         System.Int32    1   instance <Id>k__BackingField
727ef7a4 4         System.String   0   instance <Address>k__BackingField
727ee3ac 8  ...Int32, mscorlib]]   0   instance <ItemIds>k__BackingField

0:006> .foreach (obj {!dumpheap -mt 01594ddc -short}) { as /mu ${/v:address}
dwo(${obj}+4)+8; .block { .if ($spat("${address}", "*å*")) { .printf "Got it!
${address} in object %x", ${obj}; .echo }; ad /q * } }
Got it! 233 Håmpton St. in object 34f5328
```

How Does This Flow Work?

- Ever wanted to know what happens during the GC mark phase in excruciating detail?

```
0:000> bp clr!WKS::gc_heap::mark_phase

0:000> g

0:000> wt -l 1
Tracing clr!WKS::gc_heap::mark_phase to return address 7371359f
    43     0 [  0] clr!WKS::gc_heap::mark_phase
     8     0 [  1]   clr!WKS::gc_heap::generation_size
    76     8 [  0] clr!WKS::gc_heap::mark_phase
    18     0 [  1]   clr!WKS::gc_heap::generation_size
   113    26 [  0] clr!WKS::gc_heap::mark_phase
    33     0 [  1]   clr!SystemDomain::GetTotalNumSizedRefHandles
   133    59 [  0] clr!WKS::gc_heap::mark_phase
   558     0 [  1]   clr!GCToEEInterface::GcScanRoots
   138   617 [  0] clr!WKS::gc_heap::mark_phase
     8     0 [  1]   clr!WKS::fire_mark_event
   145   625 [  0] clr!WKS::gc_heap::mark_phase
  1417     0 [  1]   clr!WKS::gc_heap::scan_background_roots
...
```

```
101239 instructions were executed in 101238 events (0 from other threads)
Function Name                                  Invocations MinInst MaxInst AvgInst
clr!CORProfilerTrackGC                                   1       5       5       5
clr!GCScan::GcDhInitialScan                              1      19      19      19
clr!GCScan::GcScanHandles                                1      31      31      31
clr!GCScan::GcWeakPtrScan                                1     129     129     129
clr!GCScan::GcWeakPtrScanBySingleThread                  1      11      11      11
clr!GCToEEInterface::AfterGcScanRoots                    1      21      21      21
clr!GCToEEInterface::GcScanRoots                         1     558     558     558
clr!GcNotifications::GcNotifications                     1       6       6       6
clr!Ref_CheckAlive                                       1     127     127     127
clr!SystemDomain::GetTotalNumSizedRefHandles             1      33      33      33
clr!WKS::CFinalize::GcScanRoots                          1      27      27      27
clr!WKS::CFinalize::ScanForFinalization                  1     118     118     118
clr!WKS::fire_mark_event                                 4       8       8       8
clr!WKS::gc_heap::generation_size                        2       8      18      13
clr!WKS::gc_heap::mark_phase                             1     252     252     252
clr!WKS::gc_heap::mark_through_cards_for_large_          1   19892   19892   19892
clr!WKS::gc_heap::mark_through_cards_for_segmen          1   78241   78241   78241
clr!WKS::gc_heap::scan_background_roots                  1    1417    1417    1417
clr!WKS::gc_heap::scan_dependent_handles                 2      24      24      24
clr!WKS::gc_heap::update_card_table_bundle               1     246     246     246
```

Scripting With PyKD

- Now, some people might prefer Python to WinDbg scripts (why?!)
- Enter PyKD and windbglib

```
sp = reg("rsp")
ip = reg("rip")
while sp < stackBase:
  sym = findSymbol(ptrPtr(sp))
  if '!' in sym or '+' in sym:
    pip = ptrPtr(sp)
    psp = sp+8
    kbOutput = dbgCommand("k = %016x %016x 1000" % (psp, pip))
    if "RtlUserThreadStart" in kbOutput.split('\n')[-2]:
      dprintln("<u>Candidate call stack</u>:", True)
      dprintln(kbOutput)
      dprintln('<exec cmd="r rsp=%016x; r rip=%016x">Set RSP=%016x,
                RIP=%016x</exec>\n' % (psp, pip, psp, pip), True)
  sp += 8
```

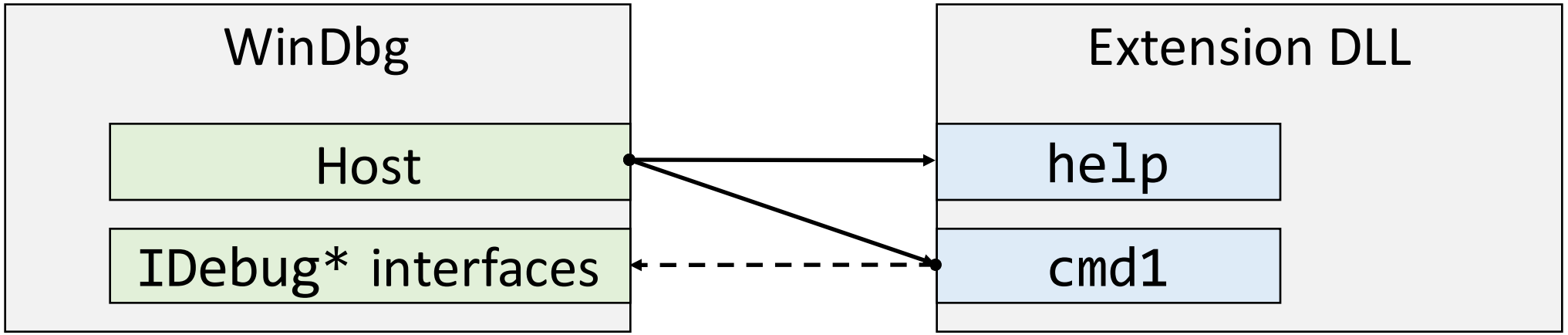
Example: heap_stat.py

```
0:001> !py heap_stat.py -stat
Running x /2 *!*`vftable' command…DONE
Running !heap -h 0 command…DONE
Enumerating 218 heap blocks
Enumerated 100 heap blocks
Enumerated 200 heap blocks

Statistics:
Type name                                  Count         Size
Payroll!employee                             100         3200
MSVCP110!std::ctype<char>                      1      Unknown
MSVCP110!std::ctype<wchar_t>                   1      Unknown
MSVCP110!std::ctype<unsigned short>            1      Unknown
MSVCR110!std::bad_alloc                        1      Unknown
Payroll!manager                                1           44
MSVCP110!std::locale::_Locimp                  1      Unknown
```

WinDbg Extensions

- WinDbg ships with a number of useful extensions
- Third-party extensions are widely available
- Simple extension model:



- You can write extensions in C#, too

Example: wbext

```
[DllExport]
public static HRESULT wb(IntPtr client,
                         [MarshalAs(UnmanagedType.LPStr)] string args)
{
  var webClient = new WebClient();
  var result = webClient.DownloadString(args);
  WriteLine(result);
  return HRESULT.S_OK;
}
```

```
0:000> .load C:\exts\wbext.dll
0:000> !wb http://example.com
…
```

x64 Strikes Back

- x64 uses a register-based calling convention (RCX, RDX, R8, R9, XMM)
- This often makes it hard to reconstruct function arguments:

```
0:000> !clrstack -a
…
OrderService.Program.WaitForMultipleObjects(UInt32, IntPtr[], Boolean, UInt32)
DomainBoundILStubClass.IL_STUB_PInvoke(UInt32, IntPtr[], Boolean, UInt32)
    PARAMETERS:
        <no data>
        <no data>
        <no data>
        <no data>
```

CMKD: Parameter Reconstruction

- CMKD uses heuristics to identify argument values

```
0:000> .load cmkd_x64.dll; !stack -p -t
## Stack-Pointer    Return-Address   Call-Site
…
01 0000002a38ffeab0 00007ffcc212c1ce KERNELBASE!WaitForMultipleObjectsEx+ef
        Parameter[0] = 000000000000001 : rcx saved in current frame into NvReg rbx
                                         which is saved by child frames
        Parameter[1] = 000001da01404418 : rdx saved in current frame into NvReg r13
                                         which is saved by child frames
        Parameter[2] = aca30f2100000001 : r8  saved in current frame into stack
        Parameter[3] = 00000000ffffffff : r9  saved in current frame into NvReg r12
                                         which is saved by child frames
…
0:000> !handle poi(000001da01404418) 8
Handle 248
  Object Specific Information
    Event Type Manual Reset
    Event is Waiting
```

SOSEX: What SOS Should Have Been

- SOSEX is an extension developed by Steve Johnson
- My favorite feature is the heap index (for large heaps)

```
0:000> !bhi
…
0:000> .foreach (obj {!dumpheap -type System.Byte[] -short}) { !mroot ${obj} }
F-Reachable queue @ 0000000000f00f38
00000000033603f8[MemoryLeak.Employee]
00000000033603f8[MemoryLeak.Schedule]
00000000033604l0[System.Byte[]]
…
0:000> !frq -stat
Freachable Queue:
       Count      Total Size   Type
-------------------------------------------------------
        6140          147360   MemoryLeak.Employee

6,140 objects, 147,360 bytes
```

Operation	Time (ms)
30 × !gcroot	≈5290
30 × !mroot	≈0

netext

- netext is an extension developed by Rodney Viana
- Designed for production troubleshooting with a strong focus on ASP.NET and WCF application diagnostics

```
0:000> .load x64\netext; !wruntime
First Request   : 12/8/2016 10:10:30 AM
Runing Time     : 00:05:37
App Pool User   : IIS APPPOOL\DefaultAppPool
Active Requests : 0n1
Path            : C:\Temp\sdpapp\SDPApp.Web\ (local disk)
...
0:000> !whttp
HttpContext    Thread Time Out Running  Status Verb     Url
000001f1651462e8   -- Not set  Finished    200 GET      http://localhost:80/
000001f2650f6698   -- Not set  00:05:45    200 NA       /SDPApp.Web/
```

More HTTP Info

```
0:000> !whttp 000001f2650f6698
Context Info
==================================
Address              : 000001f2650f6698
Target/Dump Time     : 12/8/2016 10:16:24 AM
Request Time         : 12/8/2016 10:10:29 AM
Running time         : 00:05:54
HttpContext.Items[]: 000001f16513eec0

Request Info
==================================
/SDPApp.Web/
Content Length   : -1

Response Info
==================================
Warning: Response has not completed
Status             : 200 (NULL)
Content Type       : text/html
```

netext Heap Objects Query

- A convenient SQL-like syntax is supported for finding and displaying interesting objects

```
0:000> !wfrom -type *.HttpContext
       select $addr(), _request._rawUrl, _response._statusCode
calculated: 000001F1651462E8
_request._rawUrl: /SDPApp.Web/
_response._statusCode: 0n200
calculated: 000001F2650F6698
_request._rawUrl: /SDPApp.Web/
_response._statusCode: 0n200
calculated: 000001F3651BA7E0
_request._rawUrl:
/SDPApp.Web/__browserLink/requestData/4e2517c3b6684dd3ab96b5196de99677
_response._statusCode: 0n200


3 Object(s) listed
```

Revisiting Our Earlier Example…

```
0:000> !wfrom -type OrderService.Order
       where $contains(_Address_k__BackingField, "å")
       select $addr(), _Address_k__BackingField
calculated: 000001ED5082B4D8
_Address_k__BackingField: 233 Håmpton St.

1 Object(s) listed
10,000 Object(s) skipped by filter
```

tracer

- tracer is my WinDbg extension for generic resource leak tracking (files, sockets, DB connections, bitmaps, what have you)

```
0:000> .load tracer_x86
0:000> bp kernelbase!CreateFileW "gu; !traceopen @eax 1; gc"
0:000> bp kernelbase!CloseHandle "gu; !traceclose dwo(@esp+4); gc"
0:000> g
0:000> !tracedisplay -stats
----- STACK #1 OPEN=0n12 CLOSE=0n0 OTHER=0n0 WEIGHT=0n12 -----
       KERNELBASE!GetTempFileNameW+0x1c3
       BatteryMeter!CPUInformation::CPUInformation+0x42
       BatteryMeter!TemperatureAndBatteryUpdaterThread+0x95
       KERNEL32!BaseThreadInitThunk+0x24
       ntdll!__RtlUserThreadStart+0x2f
       ntdll!_RtlUserThreadStart+0x1b
```

Remote Debugging

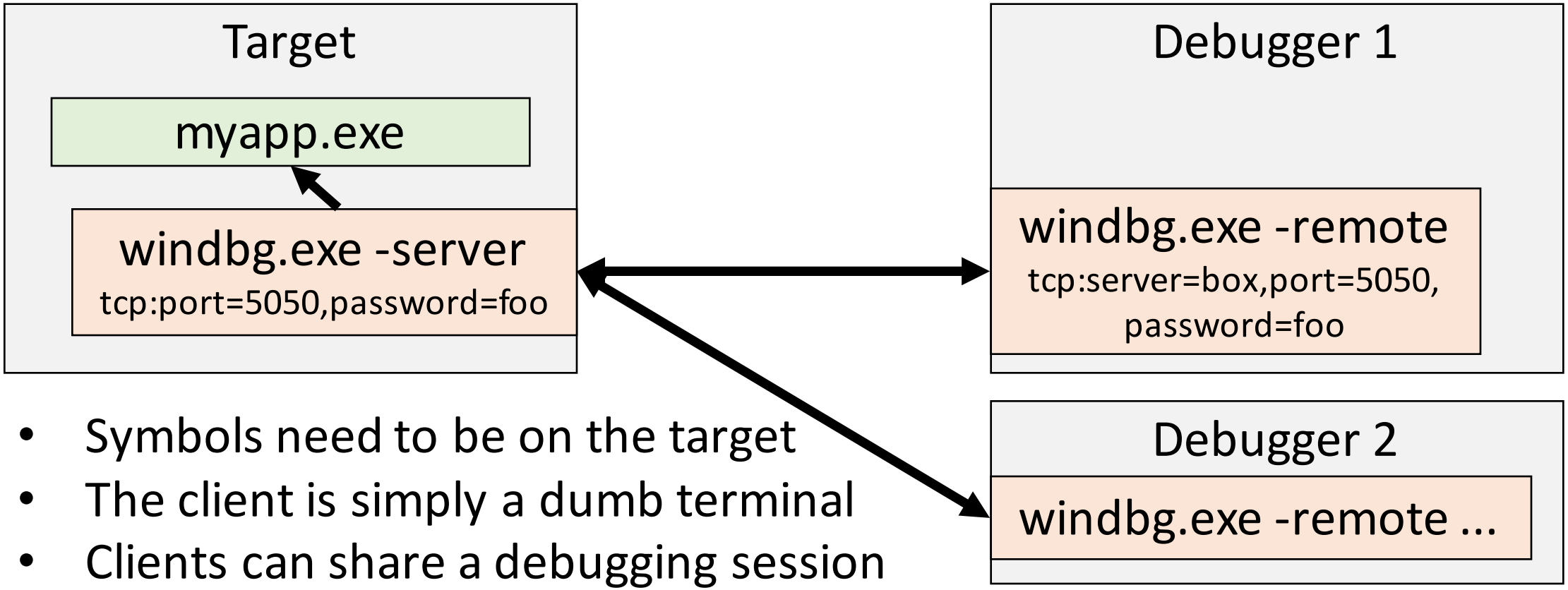
Visual Studio

- Requires Windows authentication
- Specific ports, which typically don’t travel across the Internet
- Remote debugging monitor is dumb

WinDbg (and family)

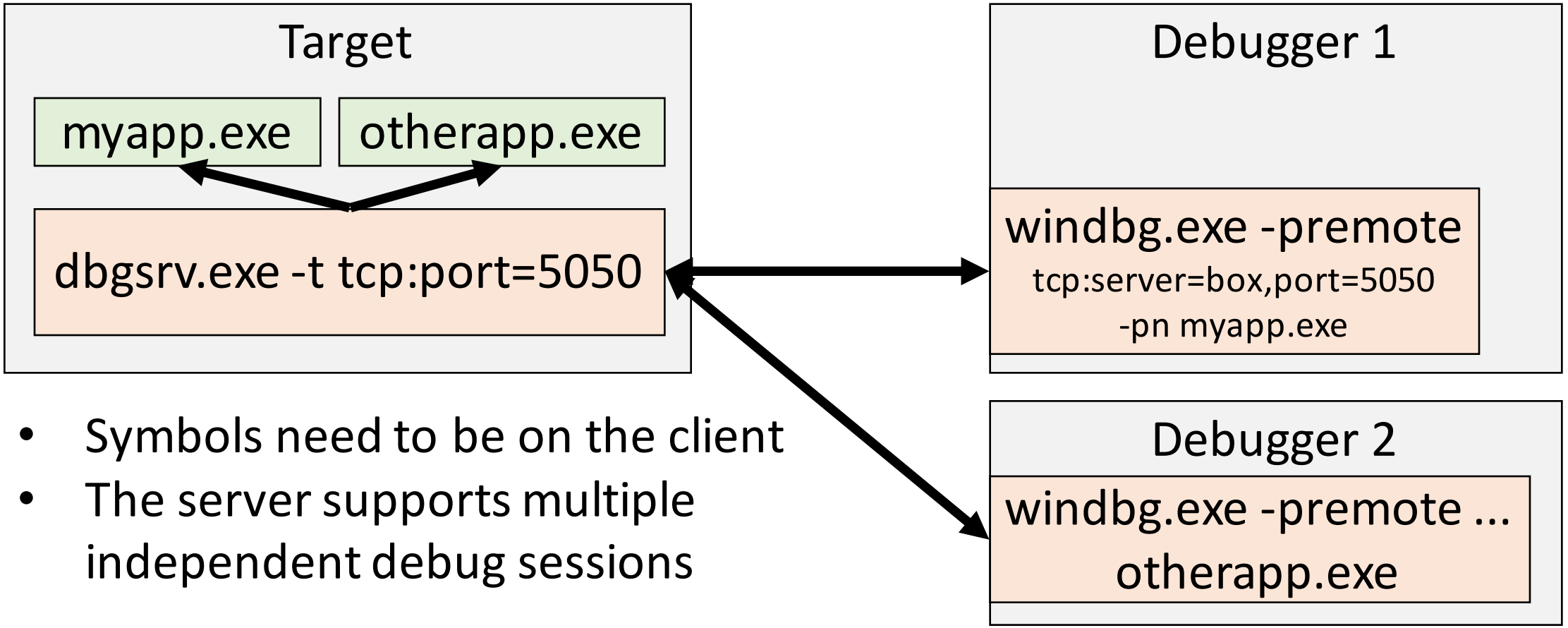
- Wide choice of transports and ports
- Does not require Windows authentication (simple password)
- Remote debugging monitor can be dumb or a full-blown debugger

WinDbg Remote Debugging Smart Server Mode



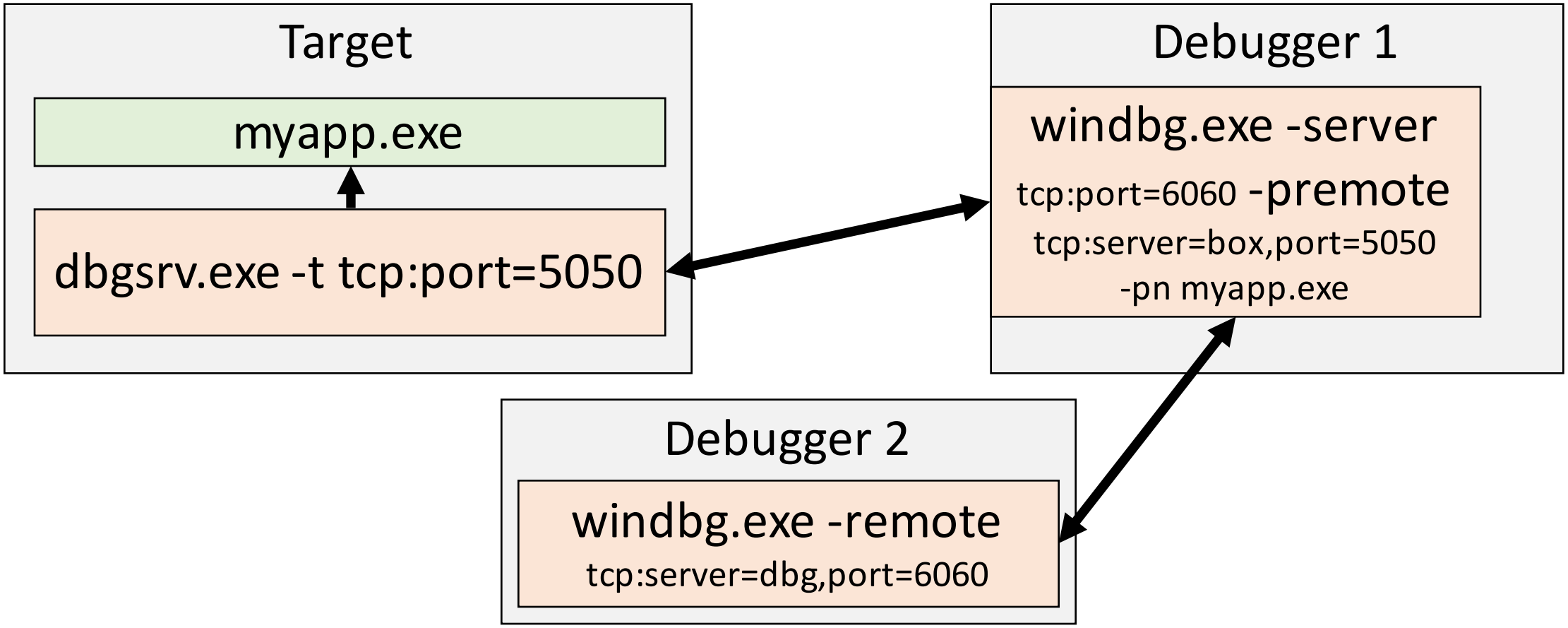
- Symbols need to be on the target
- The client is simply a dumb terminal
- Clients can share a debugging session

WinDbg Remote Debugging Smart Client Mode



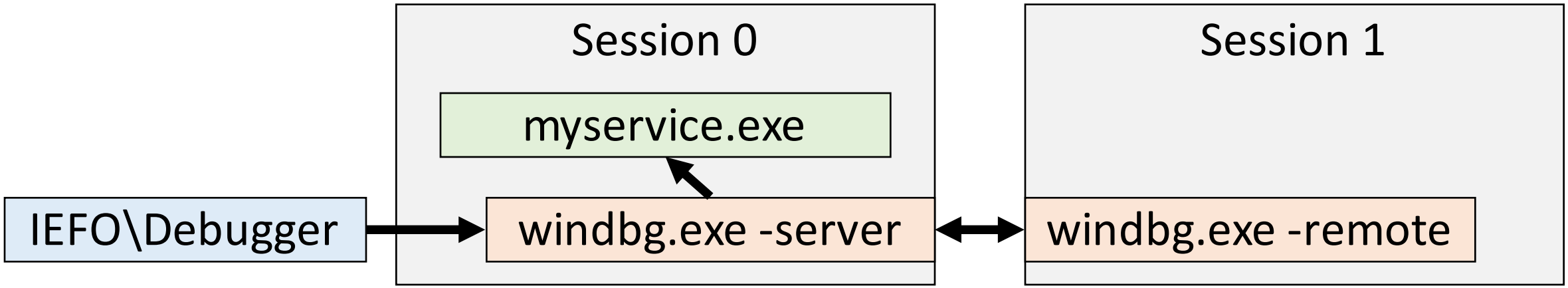
- Symbols need to be on the client
- The server supports multiple independent debug sessions

WinDbg Remote Debugging Mix And Match



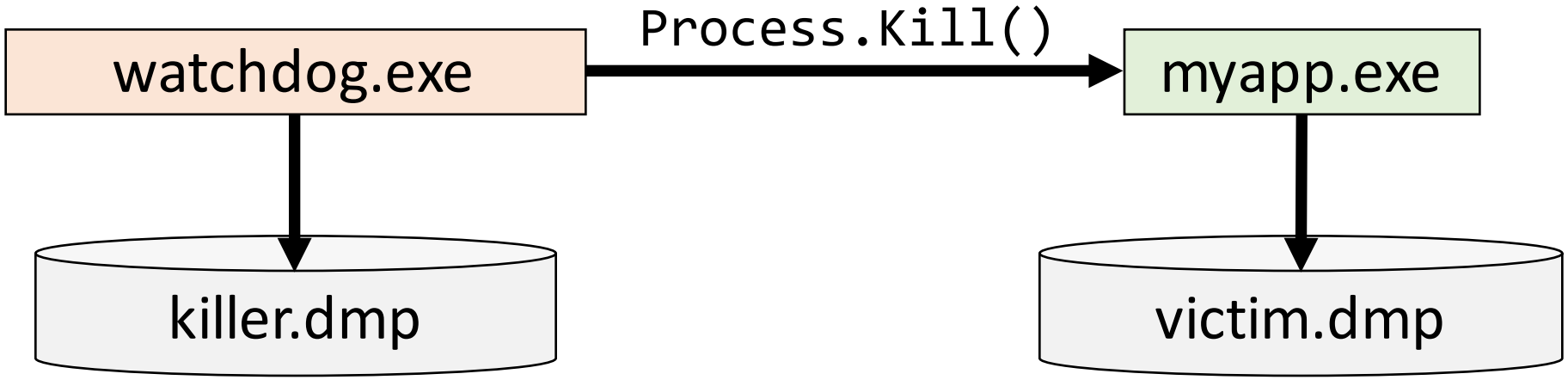
Startup Debugging

- Set **HKLM\…\IFEO\myapp.exe\Debugger** (or use GFlags) to configure startup debugger for an application
  - Especially useful with Windows services or deeply nested child processes
- For services, WinDbg cross-session-remote debugging is magical:



Apropos: Shutdown Debugging

- Occasionally, you'd have a process shut down unexpectedly without an exception
  - Terminated by someone else?
  - Calling `Environment.Exit()` or `exit()` or something similar?
- Configure silent process exit dump generation (Windows 7+) in **IFEO**



References

- My WinDbg extensions and scripts:
  - https://github.com/goldshtn/windbg-extensions
- CLRMD can replace the need for certain scripts and debugger command output parsing:
  - https://github.com/Microsoft/clrmd
- Additional extensions:
  - https://netext.codeplex.com/
  - http://www.stevestechspot.com/
  - http://www.codemachine.com/tool_cmkd.html
- msos, a CLI debugger written in C#:
  - https://github.com/goldshtn/msos

Summary

- You can keep using your toy debugger for simple bugs, but remember there are serious tools within reach if you need them
- Breakpoints at system and library functions are extremely powerful
- There are truly magical extensions out there
- Remote debugging doesn’t have to be a PITA

Thank You!

Sasha Goldshtein

@goldshtn