Easier Malware Research with WinDbg and Javascript

CARO Workshop, May 2019

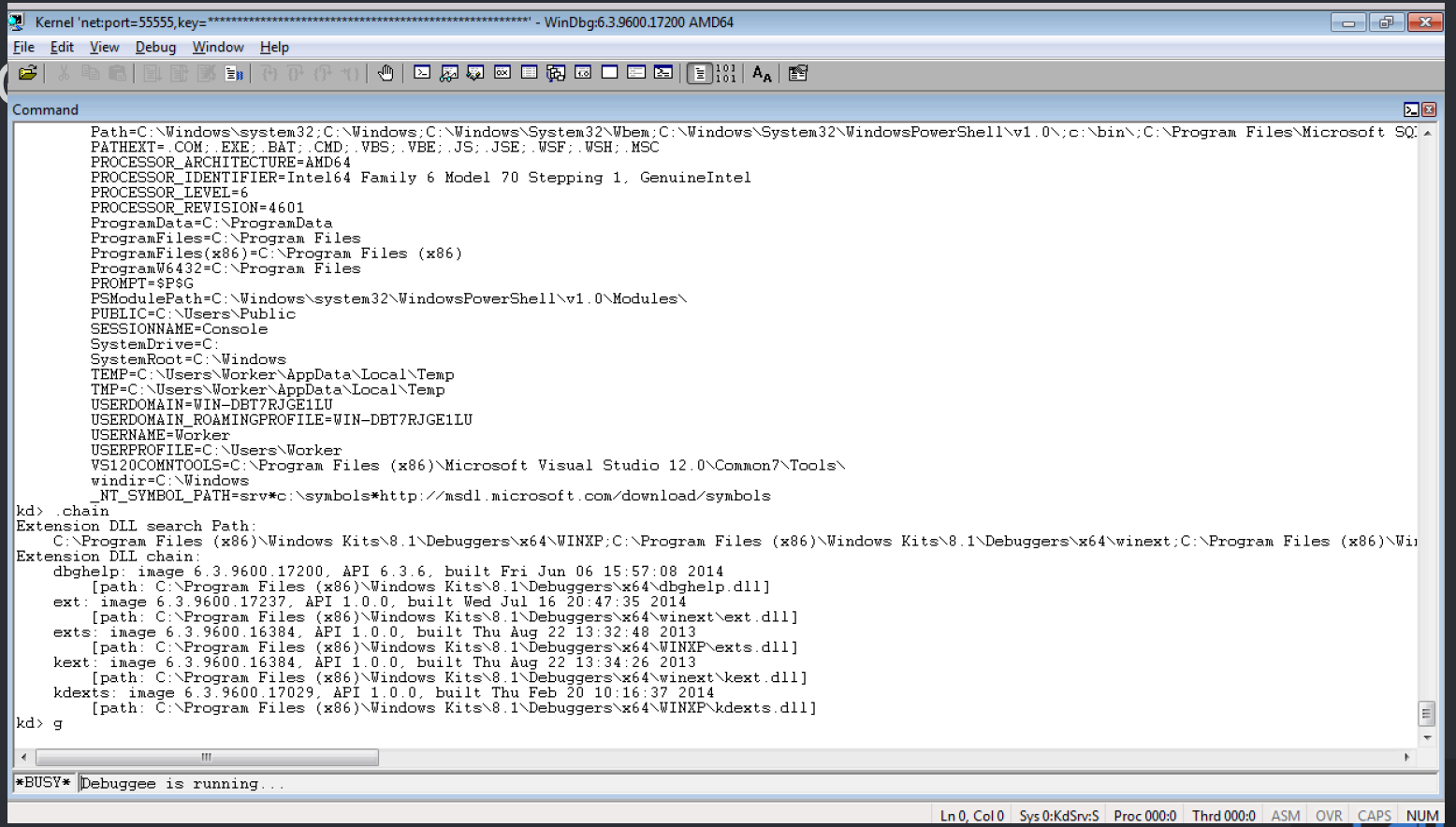
Vanja Svajcer
isvajcer@cisco.com @vanjasvajcer

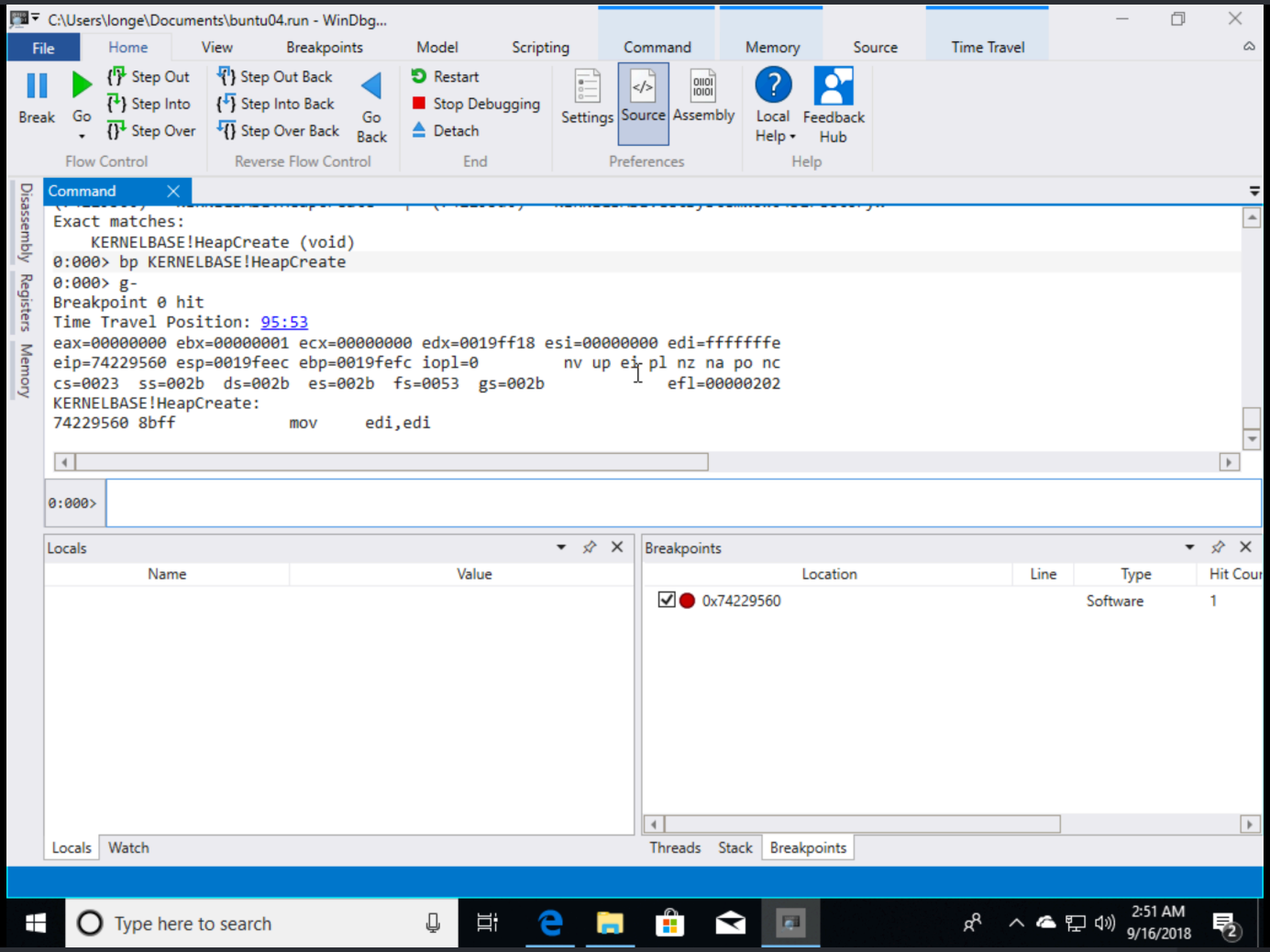
Easier WinDbg with Javascript

- Intro to WinDbg
- WinDbg scripting
    - WinDbg scripting (aka Dscript)
    - Javascript extension scripting

What is it

- (


Kernel 'net:port=55555,key=**' - WinDbg:6.3.9600.17200 AMD64
File Edit View Debug Window Help
Command
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;c:\bin\;C:\Program Files\Microsoft SQ
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 70 Stepping 1, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=4601
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=PG
PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
PUBLIC=C:\Users\Public
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Users\Worker\AppData\Local\Temp
TMP=C:\Users\Worker\AppData\Local\Temp
USERDOMAIN=WIN-DBT7RJGE1LU
USERDOMAIN_ROAMINGPROFILE=WIN-DBT7RJGE1LU
USERNAME=Worker
USERPROFILE=C:\Users\Worker
VS120COMNTOOLS=C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Tools\
windir=C:\Windows
_NT_SYMBOL_PATH=srv*c:\symbols*http://msdl.microsoft.com/download/symbols
kd> .chain
Extension DLL search Path:
C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\WINXP;C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\winext;C:\Program Files (x86)\Wi
Extension DLL chain:
dbghelp: image 6.3.9600.17200, API 6.3.6, built Fri Jun 06 15:57:08 2014
[path: C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\dbghelp.dll]
ext: image 6.3.9600.17237, API 1.0.0, built Wed Jul 16 20:47:35 2014
[path: C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\winext\ext.dll]
exts: image 6.3.9600.16384, API 1.0.0, built Thu Aug 22 13:32:48 2013
[path: C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\WINXP\exts.dll]
kext: image 6.3.9600.16384, API 1.0.0, built Thu Aug 22 13:34:26 2013
[path: C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\winext\kext.dll]
kdexts: image 6.3.9600.17029, API 1.0.0, built Thu Feb 20 10:16:37 2014
[path: C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\WINXP\kdexts.dll]
kd> g
BUSY Debuggee is running...
Ln 0, Col 0
Sys 0:KdSrv:S
Proc 000:0
Thrd 000:0
ASM
OVR
CAPS
NUM


C:\Users\longe\Documents\buntu04.run - WinDbg...
File
Home
View
Breakpoints
Model
Scripting
Command
Memory
Source
Time Travel
Break
Go
Step Out
Step Into
Step Over
Step Out Back
Step Into Back
Step Over Back
Go Back
Restart
Stop Debugging
Detach
Settings
Source
Assembly
Local Help
Feedback Hub
Flow Control
Reverse Flow Control
End
Preferences
Help
Command
Disassembly
Registers
Memory
Exact matches:
KERNELBASE!HeapCreate (void)
0:000> bp KERNELBASE!HeapCreate
0:000> g-
Breakpoint 0 hit
Time Travel Position: 95:53
eax=00000000 ebx=00000001 ecx=00000000 edx=0019ff18 esi=00000000 edi=fffffffe
eip=74229560 esp=0019feec ebp=0019fefc iopl=0 nv up ei pl nz na po nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000202
KERNELBASE!HeapCreate:
74229560 8bff mov edi,edi
0:000>
Locals
Name
Value
Locals
Watch
Breakpoints
Location
Line
Type
Hit Cou
0x74229560
Software
1
Threads
Stack
Breakpoints
Type here to search
2:51 AM
9/16/2018

Basic WinDbg

Exploration commands

- R
- x
- dt
- db, dw, dd, dq, dps, du, da
- k
- .formats
- ln - where is this?
- !dh - display pe header
- !ustr
- s

Disassembling

- u
- uf - usefull for displaying calls (uf /c)

Control

- t [address] - trace (Step into)
- p [address] - proceed (Step over)
- pc (tc) - Step over until a call instruction is encountered
- pt (tt) - Step over until return
- g - go - continue excution
- gu - go up (return to the calling function and stop - careful here)
- .process - set process context
- .thread - set register context

Breakpoints

- ba (hardware if possible)
- bp[ID] [Options] [Address [Passes]] [“CommandString"]
- bu (unresolved)
- bm (multiple)

- bl
- .bpcmds
- bc

Breakpoints

- Conditional
- `bp Address "j (Condition) 'OptionalCommands'; 'gc' "`
- `bp Address ".if (Condition) {OptionalCommands} .else {gc}”`
- `bp kernel32!CreateEventW “$$<c:\\commands.txt"`
- `bp kernel32!CreateEventW “.scriptrun c:\commands.js”`

Breakpoints

- Break on return to modify return value one liner

```
• bp kernelbase!GetTickCount "bp /1 @$ra \"
.if $ip > 00007ffea7747418 {r @$t1=@rax}
.else{r rax=@$t1 + 0x10};
r rax;g\"
;g"
```

Input/Output

- .printf
- .echo
- .readmem
- .writemem

Pseudo and temporary registers

- $csp, $ip
- $ra, $extret, $retreg
- $peb, $teb
- $proc, $thread
- $iment (operator)
- $exentry
- $t0 to $t19

Logging

- .logopen filepath
- .logclose
- .hh - open help file

Scripting

Conditional statements

- .if, .then, .else
- j (ternary) - use with conditional breakpoints
  - bp

Repetition

- .for
- .foreach
- .do
- .while
- .break
- .continue

TaLOS

Display SSDT

```
r? @$t3= *(unsigned int *) @@(nt!KiServiceLimit)
r? @$t1= (int *) @@(nt!KiServiceTable)

.for (r? @$t2=0; @$t2 < @$t3 ; r? @$t2=@$t2 + 1) {
      r? @$t4 = @$t1[@$t2] >> 4
     .printf "%y\n", @$t4 +@$t1
}
```

TALOS

Print loaded modules

```
r? @$t0 = (nt!_LIST_ENTRY *) (&@$peb->Ldr->InLoadOrderModuleList)

.for (r? @$t1=@$t0->Flink; @$t0 != @$t1; r? @$t1=@$t1->Flink)
{
    r? @$t2 = (nt!_LDR_DATA_TABLE_ENTRY *) @@(@$t1)
   .printf "%msu\n", @@c++(&@$t2->FullDllName)
}
```

TALOS

Scripting

- Invoking scripts

```
$<Filename
$><Filename
$$<Filename
$$><Filename
$$>a<Filename [arg1 arg2 arg3 ...]
```

TALOS

Javascript to the rescue!!!

TALOS

Javascript to the rescue

- Chakracore engine integrated (EC6 implementation)
- Built on top of debugger object model
- Scripting
- Visualization
- Extending the model

Exploration commands

- dx - Explore debugger object model
  - new(ish) expression evaluator command that allowes for easier exploration
  - explore WinDbg Objects
    - DML default
    - Tab completion
    - Integrated with Javascript scripting bridge

Debugger data model evaluator

```
kd> dx Debugger
Debugger
    Sessions
    Settings
    State
    Utility
kd> dx -r1 Debugger.Sessions
Debugger.Sessions
    [0x0]             : Remote KD: KdSrv:Server=@{<Local>},Trans=@{NET:Port=
kd> dx -r1 Debugger.Sessions[0]
Debugger.Sessions[0]                  : Remote KD: KdSrv:Server=@{<Local>},
    Processes
    Id                : 0
    Attributes
```

TALOS

Debugger Object model

- Debugger
- Sessions
- Processes
- Threads
- Stack
- Modules
- Handles
- Local variables
- Settings

Linq

- Language Integrated Query
- dx @$curprocess.Modules.Select(m => m.Name).Where(n => n.Contains(“maldll”))
- dx @$currsession.Processes.Count()

Linq

```
dx @$cursession.Processes.Where(p =>
p.Name.Contains(“csrss.exe”))[pid].SwitchTo()
```

```
dx @$curprocess.Io.Handles.Where(h =>
h.Type.Contains("Process")).Select(h =>
h.Object.UnderlyingObject.SeAuditProcessCreationInfo
.ImageFileName->Name)
```

TALOS

Script running/loading

- .load jsprovider.dll
- .scriptload
- .scriptrun
- .scriptunload
- .scriptlist
- .scriptproviders

Javascript entry points

- root
- invokeScript()
- initializeScript()
- uninitializeScript()
- @$scriptContents

TALOS

Javascript to the rescue!!!

TALOS

Debugger Object model accesible from JS

```
// WinDbg JavaScript sample
// function to print input string
function logme(toprint){
    host.diagnostics.debugLog(toprint + “\n");
}

function exec(cmdstr) {
host.namespace.Debugger.Utility.Control.ExecuteCommand(cmdstr);
}
```

TALOS

SSDT script in Javascript

```
function displaySSDT() {
    let
limit=host.memory.readMemoryValues(host.getModuleSymbolAddress("ntkrnlmp","KiSe
rviceLimit"),1,4);
    let servicetable=host.getModuleSymbolAddress("ntkrnlmp","KiServiceTable");

    for (var i=0;i<limit;i++){
        let addr=host.memory.readMemoryValues(servicetable.add(i*4),1,4);
        let poistr=servicetable.add(host.Int64(addr[0]).bitwiseShiftRight(4));
        let rez=host.namespace.Debugger.Utility.Control.ExecuteCommand(".printf
\"%y\n\", " + poistr.toString());
        logme(“Service "+i+" at "+poistr.toString()+" is “+rez[0]+”\n”);
    }
}
```

TALOS

Linq and Javascript - list dlls

```
let mods=host.currentProcess.Modules.Where(function (k)
{return k.Name.includes("dll")})

.Select(function (k) {return  { name: k.Name,
adder:k.BaseAddress} });

    for (var lk of mods) {
        logme(lk.name+" at "+lk.adder.toString(16));
    }
}
```

TALOS

64 bit problems

- Javascript integers only 53 bit
- Special data class Int64 and the methods
- Very annoying!

```
let OverallCallback=CallbackRoutineBlock64.bitwiseAnd(host.Int64(0x1fffffffffffff));
logme("OverallCallback:" + CallbackRoutineBlock64.toString() + ", Masked:" + OverallCallback.toString());
OverallCallback=CallbackRoutineBlock64.bitwiseAnd(host.Int64(0x3fffffffffffff));
logme("OverallCallback:" + CallbackRoutineBlock64.toString() + ", Masked:" + OverallCallback.toString());
```

```
OverallCallback:0xffffe00190a5728f, Masked:0x1fe00190a5728f
OverallCallback:0xffffe00190a5728f, Masked:0x40000000000000
```

TALOS

64 bit problems

- Javascript integers only 53 bit
- Special data class Int64 and the methods
- Very annoying!

```
//instead of let Callback = CallbackRoutineBlock & 0xfffffffffffffff0

let LowCallback=host.Int64(CallbackRoutineBlock64.getLowPart()).bitwiseAnd(0xfffffff0);
let HighCallback=host.Int64(CallbackRoutineBlock64.getHighPart()).bitwiseShiftLeft(32);
let ExBlock=HighCallback.add(LowCallback);
let Callback=host.memory.readMemoryValues(ExBlock.add(8),1,8);
```

TALOS

Other problems

- A lot of WinDbg functionality is not exposed through JS
- Workarounds possible through ExecuteCommand function

Time travel debugging - TTD

- Record a trace
- move forwards and backwards “in time”
- Set breakpoint on an API call and go backwards
- p-
- g-
- t-

TTD dx queries

```
dx @$cursession.TTD.Data.Heap().Where(h =>
h.Action.StartsWith("Alloc")).Where(h => h.Size > 0x10000)

dx @$curprocess.TTD.Events.Where(e =>
e.Type.StartsWith("ThreadCreated")

dx -r1 @$cursession.TTD.Calls("kernelbase!CreateFile*")
dx -r1 @$cursession.TTD.Calls("kernelbase!CreateFileW")
[0].Parameters
```

TALOS

Print Winsock calls

```
function ListWSCalls() {
    var wscalls=host.currentSession.TTD.Calls("WS2_32!*");
    for (var wscall of wscalls) {
        if (wscall.Function.includes(“Unknown")){
            let
rez=host.namespace.Debugger.Utility.Control.ExecuteCommand(".printf \"%y\n\",
" + wscall.FunctionAddress.toString());

            logme("Functionr:"+ rez[0]);
        }
        else {
            logme("Function:"+wscall.Function)
        }
    }
}
```

TALOS

Find memory allocations with MZ

```
var allocs=host.currentSession.TTD.Data.Heap().Where(function (h) {return
(h.Action.includes("Alloc") || h.Action.includes("Protect")) && h.Size >
0x1000;});

for (var alloc of allocs)
    var mems=
host.currentSession.TTD.Memory(alloc.Address,alloc.Address+alloc.Size,"w");

    for (var mem of mems){
        if (mem.Value.getLowPart().bitwiseAnd(0xffff) == 0x5a4d){
        logme("MZ written at "+mem.Address.toString(16)+ " from "+
mem.IP.toString(16));
        }
    }
}
```

TaLOS

Code namespace

- Code analysis functions (wrap disassembler in JS)
  - Control flow analysis
  - Basic block analysis
  - File similarity
  - Tracing data flow (sample extension by MS)

```
var disassembler =
host.namespace.Debugger.Utility.Code.CreateDisassembler();
```

Code namespace

```
function CreateBBChecksums (address)
{
   let disasm=host.namespace.Debugger.Utility.Code.CreateDisassembler();
   let bbs= disasm.DisassembleFunction(address);
   const chksumarr=[];
   for (var bb of bbs.BasicBlocks) {
       let bbstring="";
       let bblen=bb.Instructions.Count();
       for (var inst of bb.Instructions) {
          let cbs="";
          for (var cb of inst.CodeBytes)
          {
              cbs=cbs + cb.toString(16);

          }
          bbstring += cbs;
       }
       let murmurhash=murmurhash3_32(bbstring,0).toString(16);
       //logme(bbstring + ", "+ murmurhash +", "+bblen);
```

TALOS

File system access

- Native JS access to file system
  - Reading and writing files

```
var file = host.namespace.Debugger.Utility.FileSystem.CreateFile(name);
var textWriter = host.namespace.Debugger.Utility.FileSystem.CreateTextWriter(file);
textWriter.WriteLine("Hello World");
file.Close();
```

TaLOS

Helpful tips when you are lost

- for (let member in object) { logme(member);}
- for (let item of iterable) { logme(item); }
- .address for value (native) of the object
- Model tab

TaLOS

Javascript debugger within WinDbg

- .scriptload myscript.js
- .scriptdebug myscript.js
  - command syntax well known (similar to WinDbg)
  - sx, sxe eh, sxe uh
  - bp 53:0 (line:char)
  - quit and run script using
    dx @$scriptContent.FuncName()

Easier WinDbg with Javascript

- WinDbg data model is a new(ish) way for exposing internal debugger objects - lot of new functionality
- dx, Linq, TTD
- Javascript extension the best way to script the debugger model (and the debugger)
- Experiment, write scripts and extension
- Share!

Relax and breathe!

Talos

References - setup

- https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/
- https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/getting-set-up-for-debugging
- https://www.contextis.com/blog/introduction-debugging-windows-kernel-windbg
- https://reverseengineering.stackexchange.com/questions/2297/windows-kernel-debugging-on-mac-host-using-vmware-fusion#2298
- https://communities.vmware.com/docs/DOC-15691 - vm to vm over a virtual serial port VMWare Windows

References - malware analysis

- http://blog.talosintelligence.com/2017/08/windbg-and-javascript-analysis.html
- http://blog.talosintelligence.com/2017/07/unravelling-net-with-help-of-windbg.html
- https://www.gdatasoftware.com/blog/2014/06/23953-analysis-of-uroburos-using-windbg
- http://www.sekoia.fr/blog/wp-content/uploads/2016/10/Rootkit-analysis-Use-case-on-HIDEDRV-v1.6.pdf
- https://www.youtube.com/watch?v=l2ZSG_96PoM
- https://www.offensive-security.com/vulndev/fldbg-a-pykd-script-to-debug-flashplayer/

TALOS

References - Javascript and object model

- https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/dx--display-visualizer-variables-
- https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/using-linq-with-the-debugger-objects
- https://doar-e.github.io/blog/2017/12/01/debugger-data-model/
- https://blog.talosintelligence.com/2019/02/windbg-malware-analysis-with-javascript.html
- https://github.com/hugsy/windbg_js_scripts
- https://github.com/0vercl0k/windbg-scripts
- https://github.com/Microsoft/WinDbg-Samples

References - others

- http://www.zachburlingame.com/2011/12/customizing-your-windbg-workspace-and-color-scheme/ - Workspace setup
- https://github.com/vagnerpilar/windbgtree - cmdtree
- https://github.com/vallejocc/Reverse-Engineering-Arsenal/tree/master/WinDbg - WinDbg scripting 1
- https://archive.codeplex.com/?p=kdar - WinDbg scripting 2 - Archive available
- https://githomelab.ru/pykd/pykd/wikis/User%20Manual%20rus - PyKD manual - Russian only, translates OK
- http://windbg.info/download/doc/pdf/WinDbg_cmds.pdf - WinDbg commands cheatsheet
- https://www.youtube.com/watch?v=vz15OqiYYXo&feature=share - Windows Internals by Alex Sotirov
- http://terminus.rewolf.pl/terminus/ - Project Terminus Undocumented Structures Diff

TALOS

References - driver loading tools

- https://www.osronline.com/article.cfm?article=157
- http://www.novirusthanks.org/products/kernel-mode-driver-loader/
- https://github.com/maldevel/driver-loader
-

References - extensions

- https://www.microsoft.com/en-us/download/details.aspx?id=53304 - Mex
- https://github.com/comaeio/SwishDbgExt
- https://github.com/swwwolf/wdbgark
- https://githomelab.ru/pykd/pykd/wikis/Pykd%20bootstrapper - PyKD
- https://github.com/corelan/windbglib - windbglib and mona.py
- https://github.com/pstolarz/dumpext - extension for dumping PE from memory
- http://www.andreybazhan.com/dbgkit.html - Dbgkit

References - books

- Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation (Chapters 3 and 4)
- Practical Malware Analysis: A Hands-On Guide to Dissecting Malicious Software (Chapter 10)
- Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code (Chapter 14)
- The Art Of Memory Forensics – Detecting Malware and Threats in Windows, Linux and Mac Memory
- Rootkit Arsenal
- Advanced Windows Debugging
- Windows Internals
- Windows NT Device Driver Development

References - videos

- https://www.youtube.com/playlist?list=PLhx7-txsG6t6n_E2LgDGqgvJtCHPL7UFu - WinDbg tutorials by TheSourceLens
- https://www.youtube.com/watch?v=s5gOW-N9AAo&list=PLb07KvumDAnD39kssVz7DgmvNH5j89k3b Hacking Livestream #28: Windows Kernel Debugging Part I
- https://channel9.msdn.com/Shows/Defrag-Tools/Defrag-Tools-170-Debugger-JavaScript-Scripting - WinDbg JavaScript scripting
- https://channel9.msdn.com/Shows/Defrag-Tools/Defrag-Tools-138-Debugging-dx-Command-Part-1 - Dx command part 1 (and then 2)
- https://channel9.msdn.com/Shows/Defrag-Tools/Defrag-Tools-169-Debugging-Tools-for-Windows-Team - for Debugger object model
- https://www.youtube.com/watch?v=l1YJTg_A914 - Time Travel Debugging

talosintel.com
@talossecurity
@vanjasvajcer

isvajcer@cisco.com