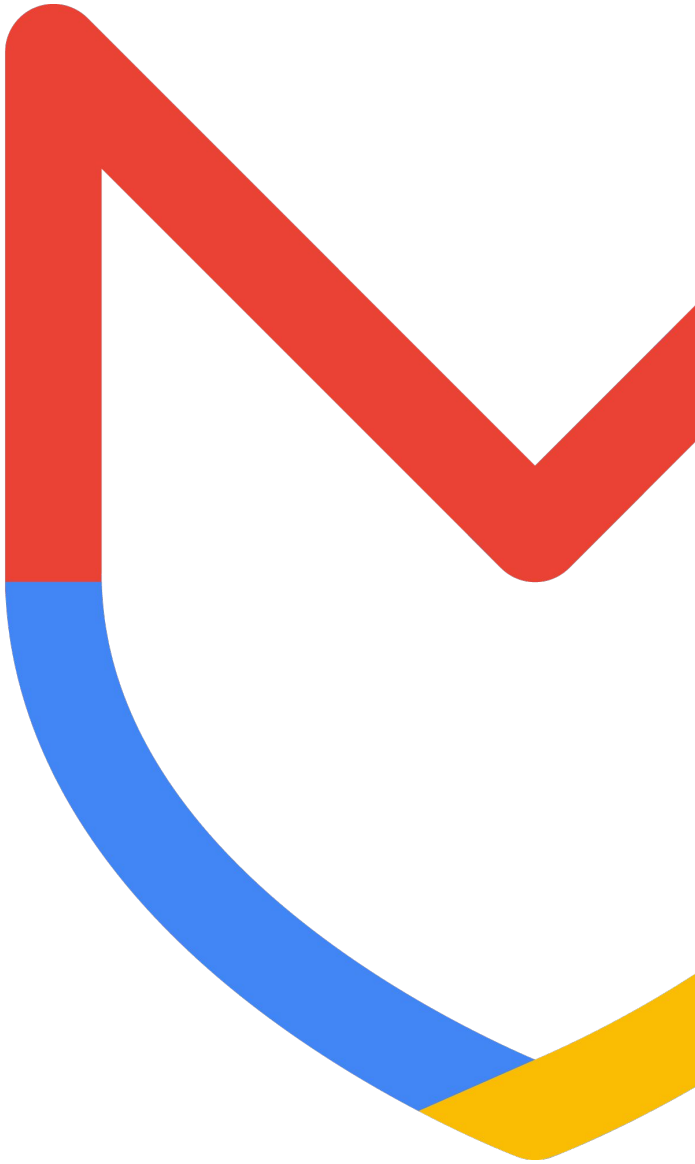
Accelerating Malware Analysis with WinDbg Time Travel Debugging (TTD)



Google Cloud
Security

Proprietary & Confidential

Course Outline

Introduction	00
Time Travel Debugging Essentials	01
Analyzing a TTD Trace	02
Practical TTD Triage	03

00 Introduction

Workshop Objectives and Logistics

Objectives

- Understand Time Travel Debugging's features and capabilities
- Learn the basics of TTD and how to incorporate it into a malware analysis workflow
- Learn how to extend TTD through JavaScript

Requirements

- Basic understanding of debuggers and x86 assembly; prior WinDbg experience not required
- FLARE-VM with the latest version of WinDbg

01
Time Travel Debugging Essentials

Why TTD for Malware Analysis?

Dramatically accelerates navigation and analysis of complex malware

- Ability to replay and step backwards through execution reduces analysis overhead (snapshots, restarts, etc.)

Highly effective against obfuscation and multi-stage malware

- Facilitates narrowing of execution events down to only those relevant for malware analysis
- Provides efficient methods for inquiring about API calls and memory access events

Enables sharing dynamic analysis with collaborators

- Records shared documentation for analyzing / extracting artifacts from a given malware family
- Complements downstream work such as automation and plugin/signature creation

Bypasses traditional anti-debug measures

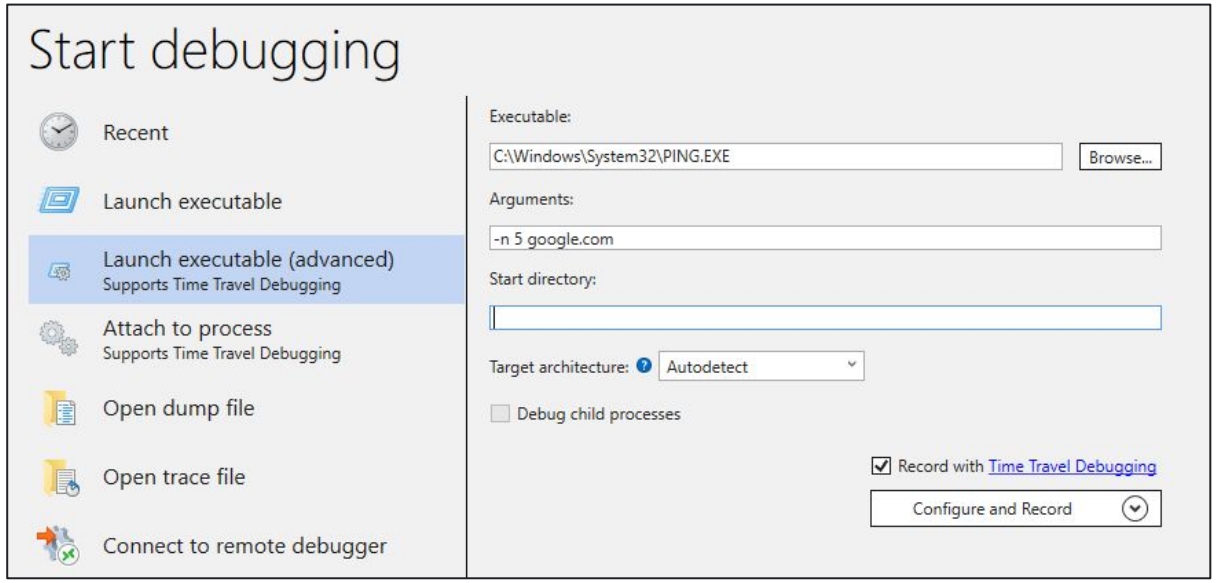
- TTD's trace recording mechanism is not the same as live debugger attachment
- Anti-debug techniques that target live debugging artifacts are ineffective against TTD

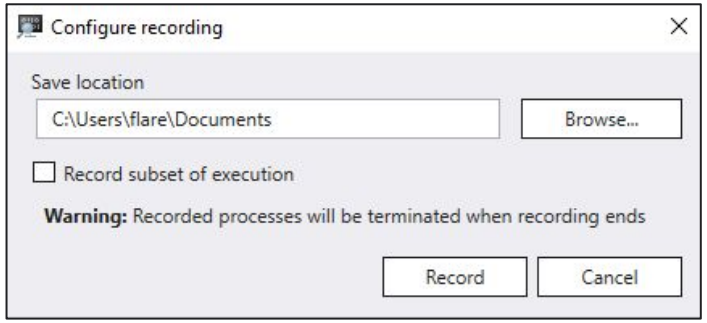
Capturing a TTD Trace: WinDbg

Navigate to `File > Launch executable (advanced)`

- Fill in the executable path, arguments, and the start directory
- Select the `Record with Time Travel Debugging` option and select `Configure and Record`
  - Note that the `Debug child processes` option is not available through the WinDbg GUI
- Enter the save location for the trace in the `Configure recording` dialog then press `Record`

⚠ Recording a trace involves execution of the selected executable. Take all usual precautions for dynamic analysis.





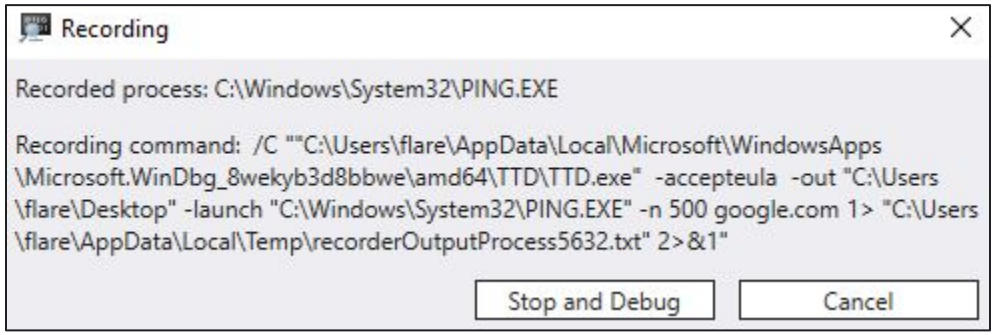
Capturing a TTD Trace: WinDbg

WinDbg records a trace until the process terminates

- Many malware do not terminate or terminate quickly – recommended to manually terminate the recording once the desired subset of execution has been captured

Recording dialog presents two options for terminating a recording manually

- `Stop and Debug` terminates the trace/process and immediately loads the trace for analysis
- `Cancel` terminates the trace/process but does not load the trace for analysis



Capturing a TTD Trace: ttd.exe

`ttd.exe` is a command-line utility to record a trace

- Usage: `ttd.exe [options] [mode] [program [<arguments>]]`

While recording a trace with `ttd.exe`:

- Output form the target executable is observable in the console
- Small menu available to terminate the recording

```
C:\Users\flare\Desktop>ttd.exe ping.exe -n 5 google.com

Launching 'ping.exe -n 5 google.com'
    Recording process (PID:6936) on trace file: C:\Users\flare\Desktop\ping01.run

Pinging google.com [142.251.41.14] with 32 bytes of data:
Reply from 142.251.41.14: bytes=32 time=3ms TTL=255
...

ping.exe(x64) (PID:6936): Process exited with exit code 0 after 4235ms
  Full trace dumped to C:\Users\flare\Desktop\ping01.run
```



Capturing Children Processes

Use `ttd.exe` with the `-children` option to record a trace of children processes

- The trace for each child process will be recorded as a separate file

```
C:\Users\flare\Desktop>ttd.exe -children powershell.exe -c "calc.exe"

Launching 'powershell.exe -c "calc.exe"'
    Recording process (PID:4324) on trace file: C:\Users\flare\Desktop\powershell01.run
    Recording process (PID:8144) on trace file: C:\Users\flare\Desktop\calc01.run
powershell.exe(x64) (PID:4324): Process exited with exit code 0 after 7500ms
  Full trace dumped to C:\Users\flare\Desktop\powershell01.run

(x64) (PID:8144): Process exited with exit code 0 after 2062ms
  Trace family nesting level is 1; Parent process ID is 4324
  Full trace dumped to C:\Users\flare\Desktop\calc01.run
```

TTD Trace Output and Limitations

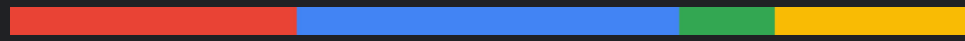
Trace output

- `<trace>.out`: information about the trace log.
- `<trace>.run`: the trace file.
- `<trace>.idx`: index file constructed from the trace when `.run` file is first opened.

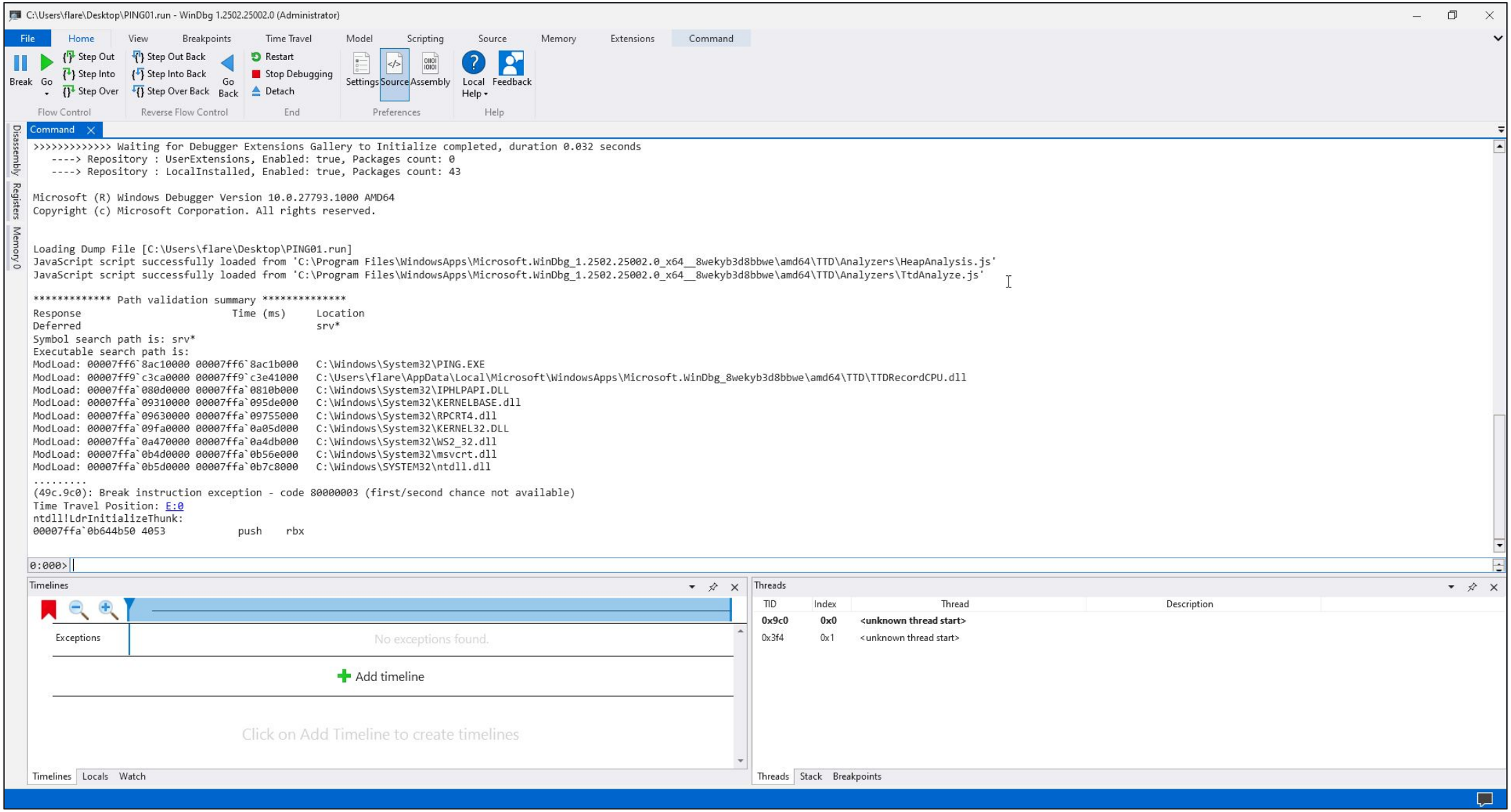
Limitations of a TTD trace

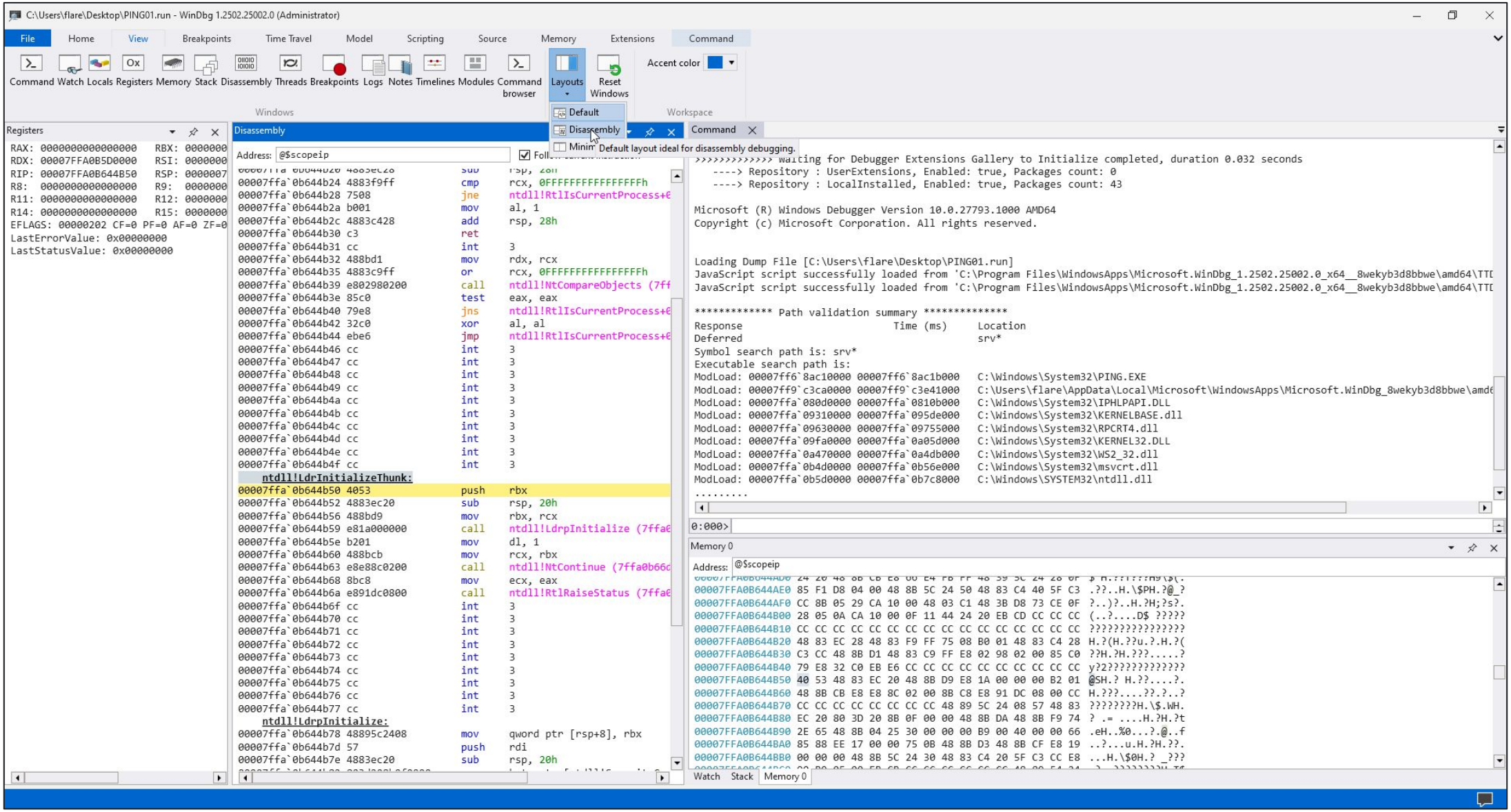
- Does not allow for any manual intervention
- Any long delays or anti-analysis techniques that prevent execution must be patched or bypassed

Exercise: Capture a Trace

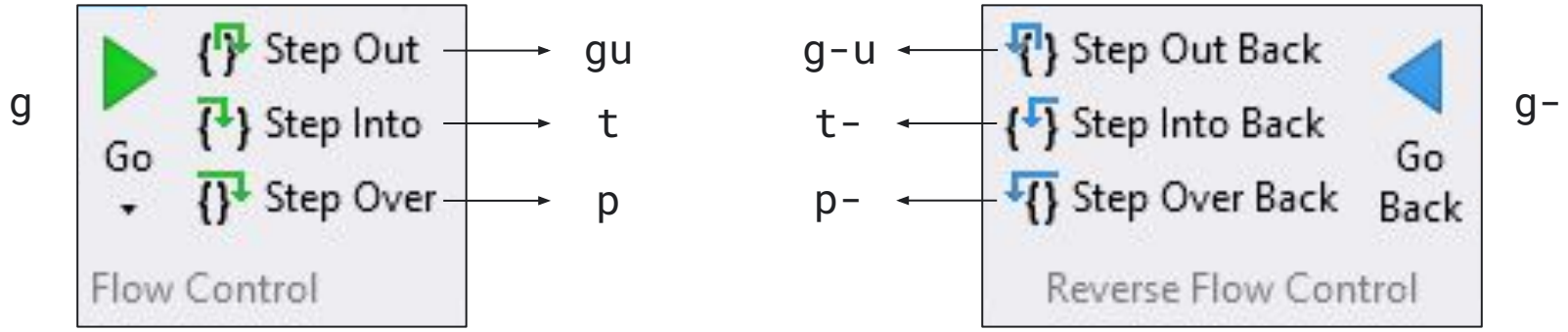


02
Analyzing a TTD Trace


C:\Users\flare\Desktop\PING01.run - WinDbg 1.2502.25002.0 (Administrator)
File
Home
View
Breakpoints
Time Travel
Model
Scripting
Source
Memory
Extensions
Command
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
Flow Control
Reverse Flow Control
End
Preferences
Help
Disassembly
Registers
Memory 0
Command
>>>>>>>>>>>>> Waiting for Debugger Extensions Gallery to Initialize completed, duration 0.032 seconds
----> Repository : UserExtensions, Enabled: true, Packages count: 0
----> Repository : LocalInstalled, Enabled: true, Packages count: 43
Microsoft (R) Windows Debugger Version 10.0.27793.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
Loading Dump File [C:\Users\flare\Desktop\PING01.run]
JavaScript script successfully loaded from 'C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2502.25002.0_x64__8wekyb3d8bbwe\amd64\TTD\Analyzers\HeapAnalysis.js'
JavaScript script successfully loaded from 'C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2502.25002.0_x64__8wekyb3d8bbwe\amd64\TTD\Analyzers\TtdAnalyze.js'
************* Path validation summary **************
Response Time (ms) Location
Deferred srv*
Symbol search path is: srv*
Executable search path is:
ModLoad: 00007ff6`8ac10000 00007ff6`8ac1b000 C:\Windows\System32\PING.EXE
ModLoad: 00007ff9`c3ca0000 00007ff9`c3e41000 C:\Users\flare\AppData\Local\Microsoft\WindowsApps\Microsoft.WinDbg_8wekyb3d8bbwe\amd64\TTD\TTDRecordCPU.dll
ModLoad: 00007ffa`080d0000 00007ffa`0810b000 C:\Windows\System32\IPHLPAPI.DLL
ModLoad: 00007ffa`09310000 00007ffa`095de000 C:\Windows\System32\KERNELBASE.dll
ModLoad: 00007ffa`09630000 00007ffa`09755000 C:\Windows\System32\RPCRT4.dll
ModLoad: 00007ffa`09fa0000 00007ffa`0a05d000 C:\Windows\System32\KERNEL32.DLL
ModLoad: 00007ffa`0a470000 00007ffa`0a4db000 C:\Windows\System32\WS2_32.dll
ModLoad: 00007ffa`0b4d0000 00007ffa`0b56e000 C:\Windows\System32\msvcrt.dll
ModLoad: 00007ffa`0b5d0000 00007ffa`0b7c8000 C:\Windows\SYSTEM32\ntdll.dll
....
(49c.9c0): Break instruction exception - code 80000003 (first/second chance not available)
Time Travel Position: E:0
ntdll!LdrInitializeThunk:
00007ffa`0b644b50 4053 push rbx
0:000>
Timelines
Exceptions
No exceptions found.
Add timeline
Click on Add Timeline to create timelines
Timelines
Locals
Watch
Threads
TID
Index
Thread
Description
0x9c0
0x0
<unknown thread start>
0x3f4
0x1
<unknown thread start>
Threads
Stack
Breakpoints


C:\Users\flare\Desktop\PING01.run - WinDbg 1.2502.25002.0 (Administrator)
File Home View Breakpoints Time Travel Model Scripting Source Memory Extensions Command
Command Watch Locals Registers Memory Stack Disassembly Threads Breakpoints Logs Notes Timelines Modules Command browser Layouts Reset Windows
Accent color
Windows
Workspace
Default
Disassembly
Minim
Default layout ideal for disassembly debugging.
Registers
RAX: 0000000000000000 RBX: 0000000
RDX: 00007FFA0B5D0000 RSI: 0000000
RIP: 00007FFA0B644B50 RSP: 0000007
R8: 0000000000000000 R9: 0000000
R11: 0000000000000000 R12: 0000000
R14: 0000000000000000 R15: 0000000
EFLAGS: 00000202 CF=0 PF=0 AF=0 ZF=0
LastErrorValue: 0x00000000
LastStatusValue: 0x00000000
Disassembly
Address: @$scopeip
00007ffa`0b644b24 4883f9ff cmp rcx, 0FFFFFFFFFFFFFFFFh
00007ffa`0b644b28 7508 jne ntdll!RtlIsCurrentProcess+0
00007ffa`0b644b2a b001 mov al, 1
00007ffa`0b644b2c 4883c428 add rsp, 28h
00007ffa`0b644b30 c3 ret
00007ffa`0b644b31 cc int 3
00007ffa`0b644b32 488bd1 mov rdx, rcx
00007ffa`0b644b35 4883c9ff or rcx, 0FFFFFFFFFFFFFFFFh
00007ffa`0b644b39 e802980200 call ntdll!NtCompareObjects (7ff
00007ffa`0b644b3e 85c0 test eax, eax
00007ffa`0b644b40 79e8 jns ntdll!RtlIsCurrentProcess+0
00007ffa`0b644b42 32c0 xor al, al
00007ffa`0b644b44 ebe6 jmp ntdll!RtlIsCurrentProcess+0
00007ffa`0b644b46 cc int 3
00007ffa`0b644b47 cc int 3
00007ffa`0b644b48 cc int 3
00007ffa`0b644b49 cc int 3
00007ffa`0b644b4a cc int 3
00007ffa`0b644b4b cc int 3
00007ffa`0b644b4c cc int 3
00007ffa`0b644b4d cc int 3
00007ffa`0b644b4e cc int 3
00007ffa`0b644b4f cc int 3
ntdll!LdrInitializeThunk:
00007ffa`0b644b50 4053 push rbx
00007ffa`0b644b52 4883ec20 sub rsp, 20h
00007ffa`0b644b56 488bd9 mov rbx, rcx
00007ffa`0b644b59 e81a000000 call ntdll!LdrpInitialize (7ffa0
00007ffa`0b644b5e b201 mov dl, 1
00007ffa`0b644b60 488bcb mov rcx, rbx
00007ffa`0b644b63 e8e88c0200 call ntdll!NtContinue (7ffa0b66c
00007ffa`0b644b68 8bc8 mov ecx, eax
00007ffa`0b644b6a e891dc0800 call ntdll!RtlRaiseStatus (7ffa0
00007ffa`0b644b6f cc int 3
00007ffa`0b644b70 cc int 3
00007ffa`0b644b71 cc int 3
00007ffa`0b644b72 cc int 3
00007ffa`0b644b73 cc int 3
00007ffa`0b644b74 cc int 3
00007ffa`0b644b75 cc int 3
00007ffa`0b644b76 cc int 3
00007ffa`0b644b77 cc int 3
ntdll!LdrpInitialize:
00007ffa`0b644b78 48895c2408 mov qword ptr [rsp+8], rbx
00007ffa`0b644b7d 57 push rdi
00007ffa`0b644b7e 4883ec20 sub rsp, 20h
Command
>>>>>>>>>>>>> Waiting for Debugger Extensions Gallery to Initialize completed, duration 0.032 seconds
----> Repository : UserExtensions, Enabled: true, Packages count: 0
----> Repository : LocalInstalled, Enabled: true, Packages count: 43
Microsoft (R) Windows Debugger Version 10.0.27793.1000 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
Loading Dump File [C:\Users\flare\Desktop\PING01.run]
JavaScript script successfully loaded from 'C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2502.25002.0_x64__8wekyb3d8bbwe\amd64\TTD
JavaScript script successfully loaded from 'C:\Program Files\WindowsApps\Microsoft.WinDbg_1.2502.25002.0_x64__8wekyb3d8bbwe\amd64\TTD
************* Path validation summary **************
Response Time (ms) Location
Deferred srv*
Symbol search path is: srv*
Executable search path is:
ModLoad: 00007ff6`8ac10000 00007ff6`8ac1b000 C:\Windows\System32\PING.EXE
ModLoad: 00007ff9`c3ca0000 00007ff9`c3e41000 C:\Users\flare\AppData\Local\Microsoft\WindowsApps\Microsoft.WinDbg_8wekyb3d8bbwe\amd6
ModLoad: 00007ffa`080d0000 00007ffa`0810b000 C:\Windows\System32\IPHLPAPI.DLL
ModLoad: 00007ffa`09310000 00007ffa`095de000 C:\Windows\System32\KERNELBASE.dll
ModLoad: 00007ffa`09630000 00007ffa`09755000 C:\Windows\System32\RPCRT4.dll
ModLoad: 00007ffa`09fa0000 00007ffa`0a05d000 C:\Windows\System32\KERNEL32.DLL
ModLoad: 00007ffa`0a470000 00007ffa`0a4db000 C:\Windows\System32\WS2_32.dll
ModLoad: 00007ffa`0b4d0000 00007ffa`0b56e000 C:\Windows\System32\msvcrt.dll
ModLoad: 00007ffa`0b5d0000 00007ffa`0b7c8000 C:\Windows\SYSTEM32\ntdll.dll
....
0:000>
Memory 0
Address: @$scopeip
00007FFA0B644AE0 85 F1 D8 04 00 48 8B 5C 24 50 48 83 C4 40 5F C3 .??..H.\$PH.?@_?
00007FFA0B644AF0 CC 8B 05 29 CA 10 00 48 03 C1 48 3B D8 73 CE 0F ?..)?..H.?H;?s?.
00007FFA0B644B00 28 05 0A CA 10 00 0F 11 44 24 20 EB CD CC CC CC (..?....D$?????
00007FFA0B644B10 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC ????????????????
00007FFA0B644B20 48 83 EC 28 48 83 F9 FF 75 08 B0 01 48 83 C4 28 H.?(H.??u.?.H.?(
00007FFA0B644B30 C3 CC 48 8B D1 48 83 C9 FF E8 02 98 02 00 85 C0 ??H.?H.???.....?
00007FFA0B644B40 79 E8 32 C0 EB E6 CC CC CC CC CC CC CC CC CC CC y?2?????????????
00007FFA0B644B50 40 53 48 83 EC 20 48 8B D9 E8 1A 00 00 00 B2 01 @SH.? H.??....?.
00007FFA0B644B60 48 8B CB E8 E8 8C 02 00 8B C8 E8 91 DC 08 00 CC H.???....??.?..?
00007FFA0B644B70 CC CC CC CC CC CC CC CC 48 89 5C 24 08 57 48 83 ????????H.\$.WH.
00007FFA0B644B80 EC 20 80 3D 20 8B 0F 00 00 48 8B DA 48 8B F9 74 ? .= ....H.?H.?t
00007FFA0B644B90 2E 65 48 8B 04 25 30 00 00 00 B9 00 40 00 00 66 .eH..%0...?.@..f
00007FFA0B644BA0 85 88 EE 17 00 00 75 0B 48 8B D3 48 8B CF E8 19 ..?...u.H.?H.??.
00007FFA0B644BB0 00 00 00 48 8B 5C 24 30 48 83 C4 20 5F C3 CC E8 ...H.\$0H.? _???
Watch Stack Memory 0

WinDbg Basics: Flow Control Panel



WinDbg Basics: Breakpoints

`bp` - Set Breakpoint

- Syntax (simplified): `[~Thread] bp [Options] [Address]`
- Sets software breakpoint at specified address

`ba` - Break on Access

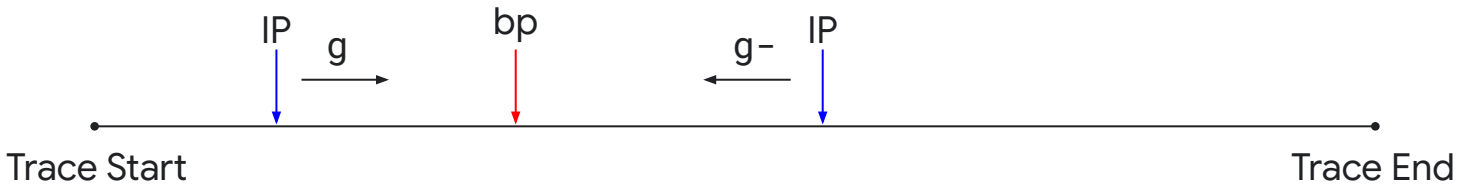
- Syntax (simplified): `[~Thread] ba Access Size [Options] [Address]`
- Sets processor (hardware) breakpoint for specified access type at specified address

`bl` - Breakpoint List

- Syntax: `bl [Breakpoints]`

`bc` - Breakpoint Clear

- Syntax: `bc Breakpoints`



Time Travel Command (`!tt`)

`!tt [position]`

- `{position}` describes a location in the trace as a:
  - Percentage (as a decimal or floating point) of trace execution - `!tt 50`
  - Timestamp (`#:#`) - `!tt 2C:1A`

`!tt br` - Break on Register

- Syntax: `!tt br[-] <register> [<value>]`
- Navigate to the previous/next position where the specified register changed values or became `<value>`

`!tt br` - Break on Access

- Syntax: `!tt ba[-] <rwe> <address> <size> [<address> <size> ...]`
- Navigate to the previous/next position where the specified memory range(s) is accessed in the specified way

`!tt br` - Break on Module

- Syntax: `!tt bm[-] [<module> ...]`
- Navigate to the previous/next module(s)

Debugger Data Model

An object-oriented hierarchy of debugger information

- Root of the hierarchy is the `Debugger` object

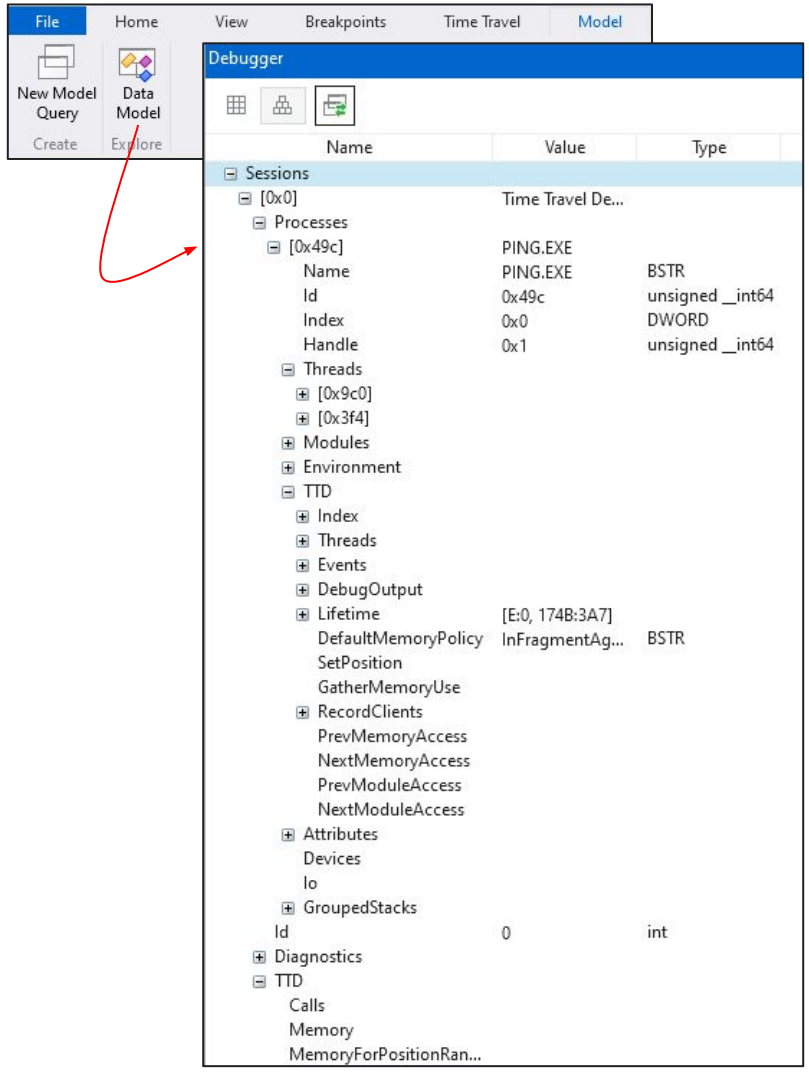
Consists of:

- Properties (e.g. function name, thread ID)
- Methods
- Containers (e.g. the list of threads in the process)

Can be accessed through:

- `dx` (Display Debugger Object Model Expression)
- Scripts written in JavaScript

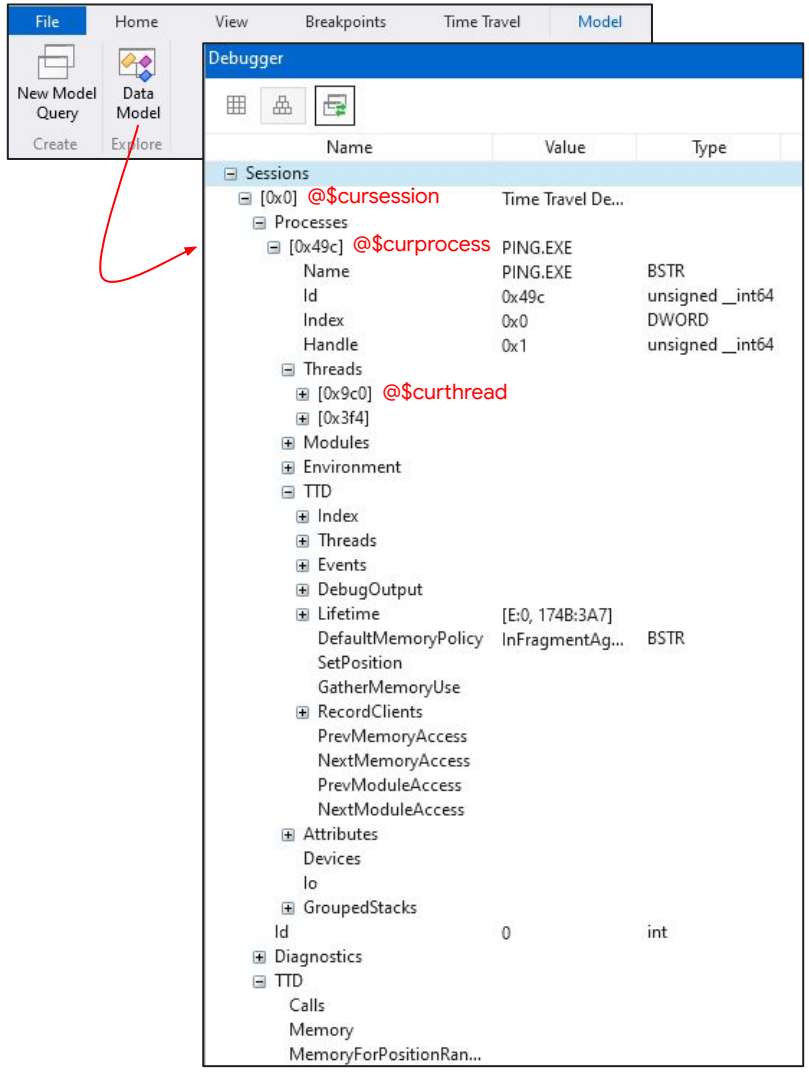
Queryable via LINQ operators



Debugger Data Model

Predefined variables:

- `@$cursession` - The current debugger session
- `@$curprocess` - The current process
- `@$curthread` - The current thread

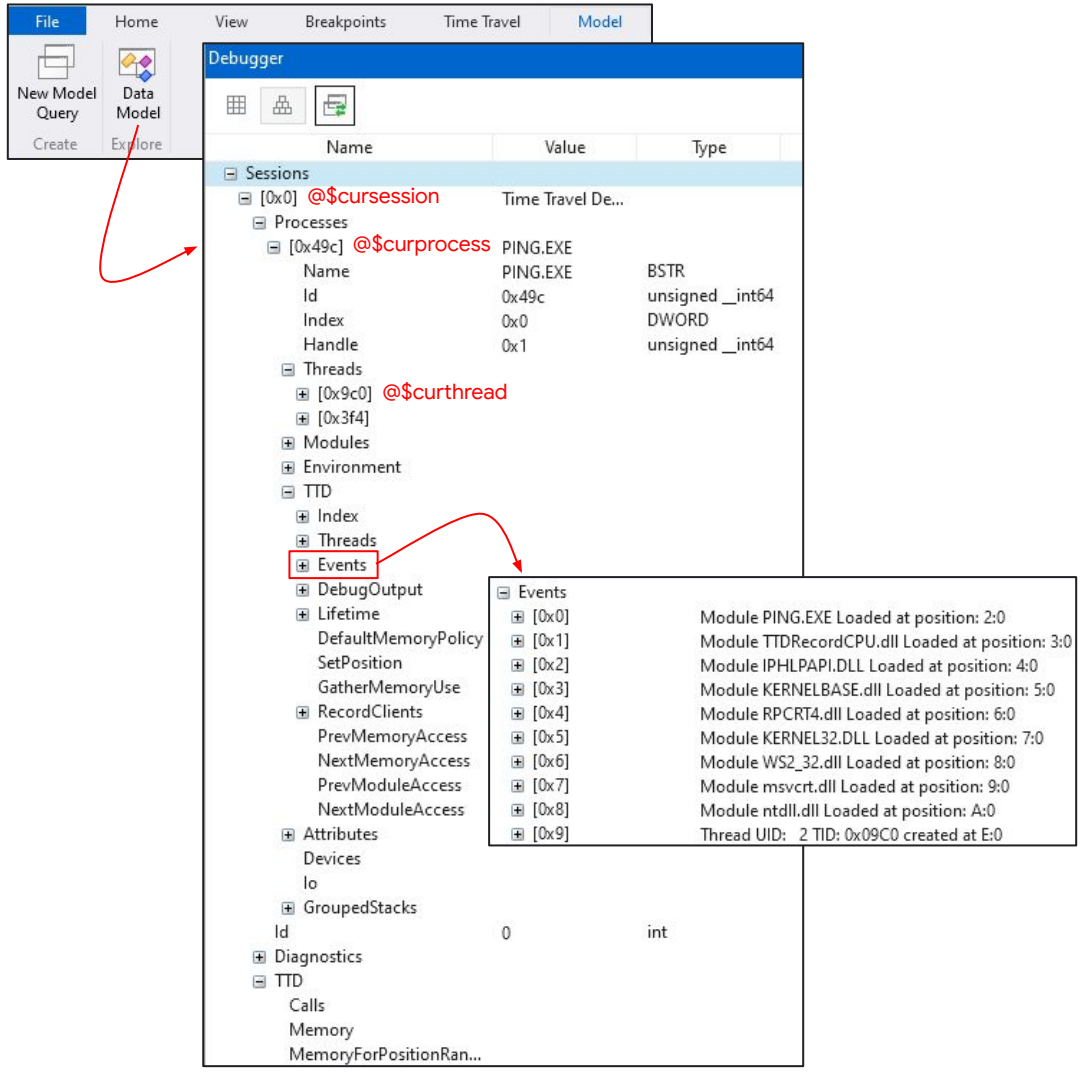


Debugger Data Model

`dx @$curprocess.TTD.Events`

- Lists events that occurred during the trace
- Provides a high-level timeline of execution history
- Event types include:
  - Exceptions
  - Thread creation and termination
  - Module load and unload

```
0:000> dx @$curprocess.TTD.Events
@$curprocess.TTD.Events
    [0x0]            : Module PING.EXE Loaded at position: 2:0
    [0x1]            : Module TTDRecordCPU.dll Loaded at position: 3:0
    [0x2]            : Module IPHLPAPI.DLL Loaded at position: 4:0
    [0x3]            : Module KERNELBASE.dll Loaded at position: 5:0
    [0x4]            : Module RPCRT4.dll Loaded at position: 6:0
    [0x5]            : Module KERNEL32.DLL Loaded at position: 7:0
    [0x6]            : Module WS2_32.dll Loaded at position: 8:0
    [0x7]            : Module msvcrt.dll Loaded at position: 9:0
    [0x8]            : Module ntdll.dll Loaded at position: A:0
    [0x9]            : Thread UID:   2 TID: 0x09C0 created at E:0
    ...              ...
```



Debugger Data Model

`dx @$curprocess.TTD.Calls("module!method1", "module!method2", ...)`

- Returns a list of call objects containing information about calls that occurred during the trace
- Allows use of wildcards, e.g. `kernel32!*`
- Effective tool for quickly locating import calls of interest

```
0:000> dx -r2 @$cursession.TTD.Calls("ws2_32!*")
@$cursession.TTD.Calls("ws2_32!*")
    [0x0]
        EventType        : 0x0
        ThreadId         : 0x9c0
        UniqueThreadId   : 0x2
        TimeStart        : 91:2C [Time Travel]
        TimeEnd          : 28D:CF [Time Travel]
        Function         : ws2_32!WSAStartup
        FunctionAddress  : 0x7ffa0a47eb10
        ReturnAddress    : 0x7ff68ac11d80
        ReturnValue      : 0x0
        Parameters
    ...
```

Debugger Data Model

```
dx @$curprocess.TTD.Calls("module!method1", "module!method2", ...)
```

- Returns a list of call objects containing information about calls that occurred during the trace
- Allows use of wildcards, e.g. `kernel32!*`
- Effective tool for quickly locating import calls of interest

Display results as a data grid

```
dx -g @$cursession.TTD.Calls("ws2_32!*")
```

	(+) EventType	(+) ThreadId	(+) UniqueThreadId	(+) TimeStart	(+) TimeEnd	(+) Function	(+) FunctionAddress	(+) ReturnAddress	(+) ReturnValue	(+) Parameters
[0x0]	0x0	0x9c0	0x2	91:2C	28D:CF	ws2_32!WSAStartup	0x7ffa0a47eb10	0x7ff68ac11d80	0x0	{...}
[0x1]	0x0	0x9c0	0x2	27B:2E	28B:5E5	WS2_32!WahCreateHandleContextTable	0x7ffa0a47f3b0	0x7ffa0a47ee81	0x0	{...}
[0x2]	0x0	0x9c0	0x2	28B:917	28B:923	WS2_32!WahOpenApcHelper	0x7ffa0a47f2f0	0x7ffa0a47f2a6	0x0	{...}
[0x3]	0x0	0x9c0	0x2	28B:949	28D:16	WS2_32!WahOpenCurrentThread	0x7ffa0a47f1c0	0x7ffa0a47f189	0x0	{...}
[0x4]	0x0	0x9c0	0x2	295:1D1	295:51A	WS2_32!GetAddrInfoW	0x7ffa0a475b20	0x7ff68ac1115c	0x2af9	{...}
[0x5]	0x0	0x9c0	0x2	295:523	1038:D62	WS2_32!GetAddrInfoW	0x7ffa0a475b20	0x7ff68ac111dc	0x0	{...}
[0x6]	0x0	0x9c0	0x2	295:7A0	325:70F	WS2_32!socket	0x7ffa0a4755f0	0x7ffa0a472d42	0x108	{...}
[0x7]	0x0	0x9c0	0x2	295:7C8	325:70A	WS2_32!WSASocketW	0x7ffa0a4756b0	0x7ffa0a47565e	0x108	{...}
[0x8]	0x0	0x9c0	0x2	2E7:237	2E7:897	WS2_32!WahCreateHandleContextTable	0x7ffa0a47f3b0	0x7ffa083e993e	0x0	{...}
[0x9]	0x0	0x9c0	0x2	321:37	321:44E	WS2_32!WahInsertHandleContext	0x7ffa0a472800	0x7ffa083e6b7b	0x1e8d8371700	{...}
[0xa]	0x0	0x9c0	0x2	325:25D	325:676	WS2_32!WahInsertHandleContext	0x7ffa0a472800	0x7ffa0a472786	0x1e8d8371360	{...}
[0xb]	0x0	0x9c0	0x2	325:714	327:4D5	WS2_32!closesocket	0x7ffa0a475000	0x7ffa0a472d5a	0x0	{...}

Debugger Data Model

`dx @$cursession.TTD.Memory(startAddress, endAddress [, "rwec"])]`

- Returns memory access information for specified address range
- Use with access mask `e` to find the first executed instruction (entrypoint) of a range of memory
- Use with access mask `w` to find the first or last time a range of memory is written to

```
0:000> dx -r1 @$curprocess.TTD.Events[0]
@$curprocess.TTD.Events[0]                 : Module PING.EXE Loaded at position: 2:0
    Type             : ModuleLoaded
    Position         : 2:0 [Time Travel]
    Module           : Module C:\Windows\System32\PING.EXE at address 0X7FF68AC10000 with size 45056

0:000> dx @$cursession.TTD.Memory(0x7FF68AC10000, 0x7FF68AC10000 + 45056, "e")[0]
@$cursession.TTD.Memory(0x7FF68AC10000, 0x7FF68AC10000 + 45056, "e")[0]
    EventType        : 0x1
    ThreadId         : 0x9c0
    UniqueThreadId   : 0x2
    TimeStart        : 16:28 [Time Travel]
    TimeEnd          : 16:28 [Time Travel]
    AccessType       : Execute
    IP               : 0x7ff68ac135f0 == $exentry
    Address          : 0x7ff68ac135f0
    Size             : 0x4
    Value            : 0x28ec8348
    SystemTimeStart  : Friday, June 27, 2025 13:22:52.202
    SystemTimeEnd    : Friday, June 27, 2025 13:22:52.202
```

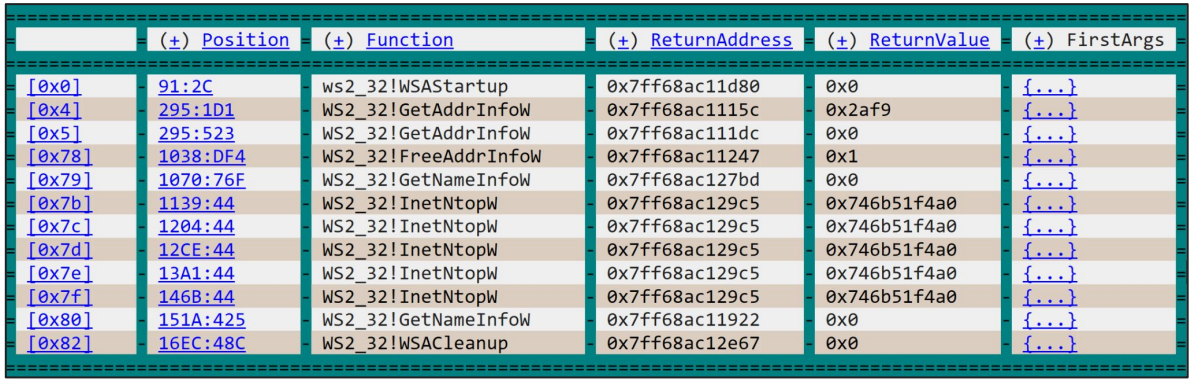
Debugger Data Model

Use LINQ queries on iterable objects to filter and manipulate debugger objects

Frequently used query methods:

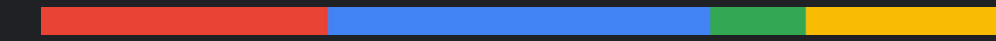
- `.First` / `.Last` - Return the first/last element in the collection
- `.Where(PredicateMethod)` - Return objects for which the predicate method returns true
- `.Select(KeyProjectorMethod)` - Return the results of calling the projector method on each object

```
0:000> dx @$moduleStart = 0x7ff68ac10000

0:000> dx @$moduleEnd = 0x7ff68ac10000 + 0xb000

0:000> dx @$calls = @$cursession.TTD.Calls("ws2_32!*")

0:000> dx @$moduleCalls = @$calls.Where(c => c.ReturnAddress > @$moduleStart
&& c.ReturnAddress < @$moduleEnd)

0:000> dx -g @$moduleCalls.Select(c => new{ Position = c.TimeStart, Function
= c.Function, ReturnAddress = c.ReturnAddress, ReturnValue = c.ReturnValue,
FirstArgs = c.Parameters})
```



	(+) Position	(+) Function	(+) ReturnAddress	(+) ReturnValue	(+) FirstArgs
[0x0]	91:2C	ws2_32!WSAStartup	0x7ff68ac11d80	0x0	{...}
[0x4]	295:1D1	WS2_32!GetAddrInfoW	0x7ff68ac1115c	0x2af9	{...}
[0x5]	295:523	WS2_32!GetAddrInfoW	0x7ff68ac111dc	0x0	{...}
[0x78]	1038:DF4	WS2_32!FreeAddrInfoW	0x7ff68ac11247	0x1	{...}
[0x79]	1070:76F	WS2_32!GetNameInfoW	0x7ff68ac127bd	0x0	{...}
[0x7b]	1139:44	WS2_32!InetNtopW	0x7ff68ac129c5	0x746b51f4a0	{...}
[0x7c]	1204:44	WS2_32!InetNtopW	0x7ff68ac129c5	0x746b51f4a0	{...}
[0x7d]	12CE:44	WS2_32!InetNtopW	0x7ff68ac129c5	0x746b51f4a0	{...}
[0x7e]	13A1:44	WS2_32!InetNtopW	0x7ff68ac129c5	0x746b51f4a0	{...}
[0x7f]	146B:44	WS2_32!InetNtopW	0x7ff68ac129c5	0x746b51f4a0	{...}
[0x80]	151A:425	WS2_32!GetNameInfoW	0x7ff68ac11922	0x0	{...}
[0x82]	16EC:48C	WS2_32!WSACleanup	0x7ff68ac12e67	0x0	{...}

Demo/Lab



03
Practical TTD Triage

TTD Triage Workflow

Step 1: `dx @$curprocess.TTD.Events`

- Review key events captured during the trace
  - Observe which modules are loaded (capabilities) as well as when they are loaded (import resolution by subsequent stages)
- Find the primary executable and save the module boundaries into variables

```
0:000> dx @$curprocess.TTD.Events
@$curprocess.TTD.Events
    [0x0]            : Module encStrings.exe Loaded at position: 2:0
    [0x1]            : Module TTDRecordCPU.dll Loaded at position: 3:0
    [0x2]            : Module KERNEL32.DLL Loaded at position: 4:0
    ...              ...
0:000> dx -r1 @$curprocess.TTD.Events[0].Module
@$curprocess.TTD.Events[0].Module                : Module C:\Users\flare\Desktop\encStrings.exe at address 0X400000 with size 53248
    Name             : C:\Users\flare\Desktop\encStrings.exe
    Address          : 0x400000
    Size             : 0xd000
    Checksum         : 0x0
    Timestamp        : 0x4feb5c75

0:000> dx @$moduleStart = @$curprocess.TTD.Events[0].Module.Address
@$moduleStart = @$curprocess.TTD.Events[0].Module.Address : 0x400000

0:000> dx @$moduleEnd = @$moduleStart + @$curprocess.TTD.Events[0].Module.Size
@$moduleEnd = @$moduleStart + @$curprocess.TTD.Events[0].Module.Size : 0x40d000
```

TTD Triage Workflow

Step 2: `dx @$cursession.TTD.Calls("module!*", ...)`

- Analyze API calls to modules of interest from the executable
- Filter and reduce results using `Where` and `Select` queries

```
0:000> dx @$calls = @$cursession.TTD.Calls("kernel32!*", "advapi32!*", "ws2_32!*", "wininet!*")
0:000> dx @$moduleCalls = @$calls.Where( c => c.ReturnAddress > @$moduleStart && c.ReturnAddress < @$moduleEnd)
0:000> dx -g @$moduleCalls.Select(c => new {Position = c.TimeStart, Function = c.Function, ReturnAddress = c.ReturnAddress,
ReturnValue = c.ReturnValue, FirstArgs = c.Parameters})
```

	(+) Position	(+) Function	(+) ReturnAddress	(+) ReturnValue	(+) FirstArgs
[0x2]	12:5F	KERNEL32!GetVersion	0x4020eb	0x23f00206	{...}
[0x3]	12:177	KERNEL32!HeapCreate	0x40362d	0x720000	{...}
[0x4]	16:5C2	KERNEL32!GetVersionExA	0x4034f3	0x1	{...}
[0x5]	18:412	KERNEL32!GetStartupInfoA	0x403355	0x19fefc00000000	{...}
⋮					
[0x44]	2D9:205B	ws2_32!WSAStartup	0x401818	0x8c5ba800000000	{...}
[0x52]	497:31D	KERNEL32!Sleep	0x401866	0x0	{...}
[0x53]	499:1A	ws2_32!gethostbyname	0x401871	0x8cef28	{...}
[0x187]	1417:2F	ws2_32!inet_ntoa	0x40189b	0x7b008c5d54	{...}
[0x189]	1417:AC	ws2_32!gethostname	0x4018c0	0x100000000	{...}
[0x1a3]	142B:B73	Wininet!InternetOpenA	0x4018db	0xcc0004	{...}
[0x2ec]	25D6:328	Wininet!InternetConnectA	0x40192d	0xcc0008	{...}
[0x341]	28D8:2E8	KERNEL32!GetTickCount	0x401987	0x223b50223b5be	{...}
[0x342]	28D8:987	Wininet!HttpOpenRequestA	0x4019c4	0xcc000c	{...}
[0x358]	299E:10F	Wininet!HttpSendRequestA	0x4019ed	0x1	{...}
[0x5f4]	3635:14C	Wininet!InternetReadFile	0x401a15	0x1	{...}
[0x61e]	3643:23DE	KERNEL32!VirtualAlloc	0x401afb	0x4160000	{...}

TTD Triage Workflow

Step 2: `dx @$cursession.TTD.Calls("module!*", ...)`

- Analyze API calls to modules of interest from the executable
- Filter and reduce results using `Where` and `Select` queries




```
0:000> !tt 3643:23DE
Setting position: 3643:23DE
(1e8c.1a80): Break instruction exception - code 80000003 (first/second chance not available)
Time Travel Position: 3643:23DE
...
KERNEL32!VirtualAlloc:
7671f3c0 8bff            mov     edi,edi

0:000> dd /c 1 esp+0x4 L4
0019ea44  00000000
0019ea48  00000400 -> dwSize
0019ea4c  00001000 -> flAllocationType = MEM_COMMIT
0019ea50  00000040 -> flProtect = PAGE_EXECUTE_READWRITE

0:000> g 0x401afb
Time Travel Position: 3645:8
eax=04160000 ebx=0031a000 ecx=00000000 edx=00000000 esi=004020bf edi=0019eb5c
eip=00401afb esp=0019ea54 ebp=0019ff30 iopl=0         nv up ei pl zr na pe nc
```

```
LPVOID VirtualAlloc(
  [in, optional] LPVOID lpAddress,
  [in]           SIZE_T dwSize,
  [in]           DWORD  flAllocationType,
  [in]           DWORD  flProtect
);
```

TTD Triage Workflow

Step 3: `dx @$cursession.TTD.Memory(startAddress, endAddress [, "rwec"])`

- Analyze memory allocations to identify when they are written to / executed

```
0:000> dx @$allocStart = 0x4160000
0:000> dx @$allocEnd = 0x4160000 + 0x400

0:000> dx @$cursession.TTD.Memory(@$allocStart, @$allocEnd, "e").First()
@$cursession.TTD.Memory(@$allocStart, @$allocEnd, "e").First()
    EventType        : 0x1
    ThreadId         : 0x1a80
    UniqueThreadId   : 0x2
    TimeStart        : 3645:12E [Time Travel]
    TimeEnd          : 3645:12E [Time Travel]
    AccessType       : Execute
    IP               : 0x4160000
    Address          : 0x4160000
    Size             : 0x1
    Value            : 0x90

0:000> !tt 3645:12E

0:000> .writemem C:\Users\flare\Desktop\shellcode.bin 0x4160000 0x4160400 -1
Writing 400 bytes
```

JavaScript Automation

Automate and extend functionality through JavaScript and debugger objects

- `.scriptload ScriptFile`: Load script. Invokes `initializeScript` function.
- `.scriptrun ScriptFile`: Run script. Invokes the `initializeScript` and `invokeScript` functions.
- `.scriptunload ScriptFile`: Unload the specified script.

Debugger objects and utility functionality are available under the `host` object

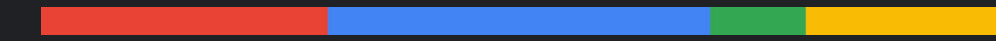
- `host.diagnostics.debugLog`: Print to debug console
- `host.memory`: Memory-related functionality (`readMemoryValues`, `readString`, `readWideString`)
- `host.currentProcess`, `host.currentSession`, `host.currentThread`

JavaScript Automation

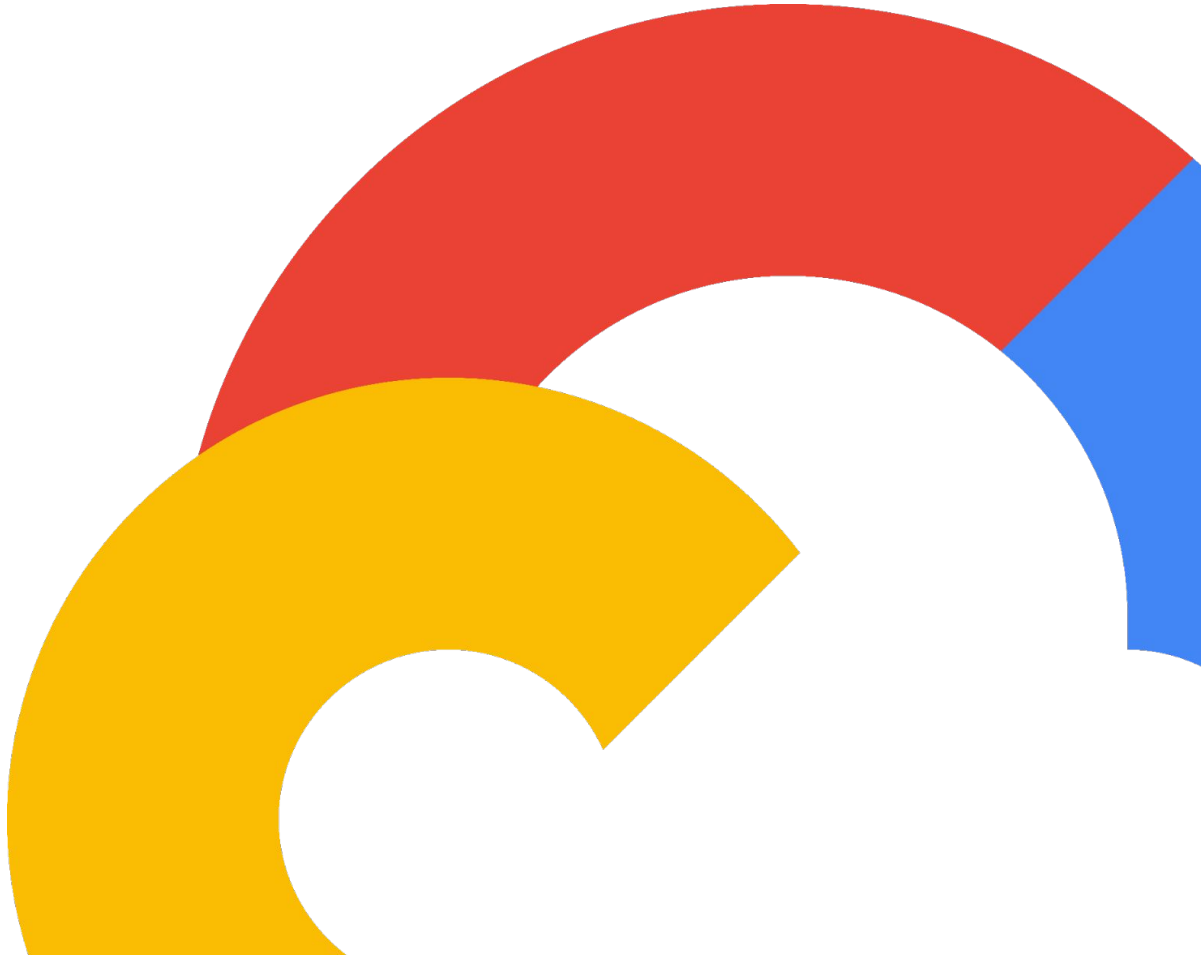
```
'use strict';
function initializeScript() {
 return [new host.functionAlias(getHostNames, 'ghn')];
}
function getHostNames(startAddr, endAddr) {
 var connectCalls = host.currentSession.TTD.Calls('wininet!InternetConnectA').Where(c => c.ReturnAddress > startAddr && c.ReturnAddress < endAddr);
 var i = 1;
 for (var c of connectCalls) {
   c.TimeStart.SeekTo();
   var hostNamePtr = host.memory.readMemoryValues(host.currentThread.Registers.User.esp + 8, 1, 4)[0];
   var hostName = host.memory.readString(hostNamePtr);
   host.diagnostics.debugLog(`${ i }. Found: ${ hostName }\n`);
   i++;
 }
}
```

```
0:000> .scriptload C:\Users\flare\Desktop\getHostNames.js
JavaScript script successfully loaded from 'C:\Users\flare\Desktop\getHostNames.js'
0:000> !ghn 0x400000, 0x40d000
(1e8c.1a80): Break instruction exception - code 80000003 (first/second chance not available)
Time Travel Position: 25D6:328
1. Found: zoidberg.mandiant.com
@$ghn(0x400000, 0x40d000)
```

Demo/Lab



Thank you



Google Cloud