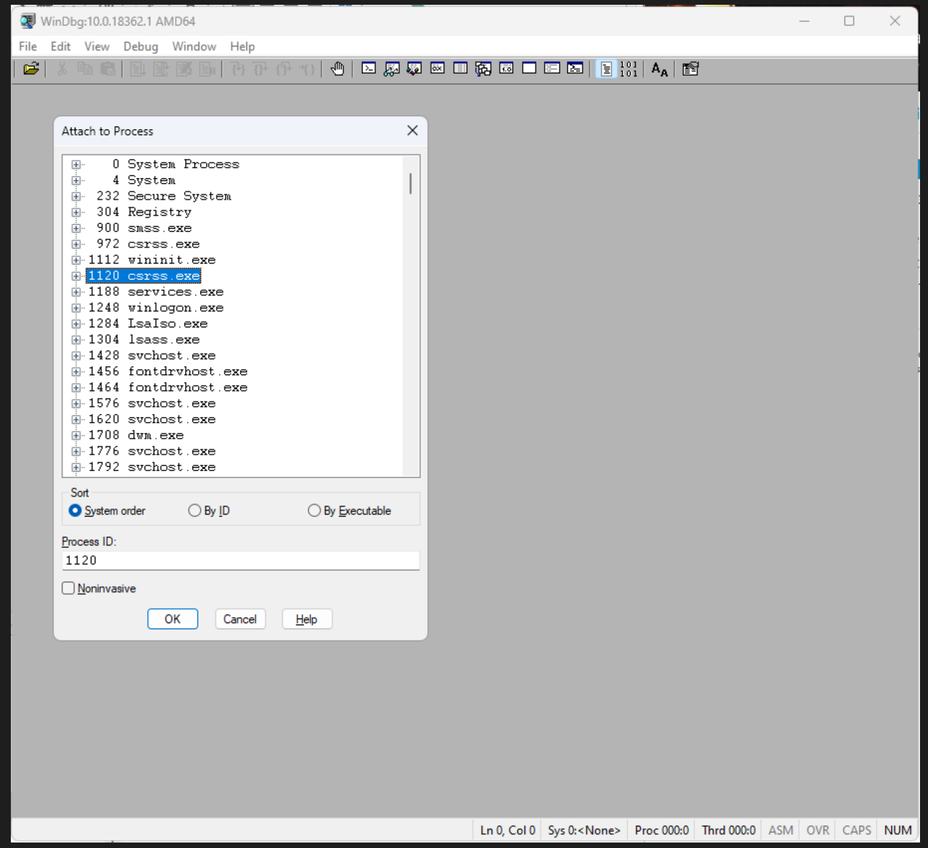
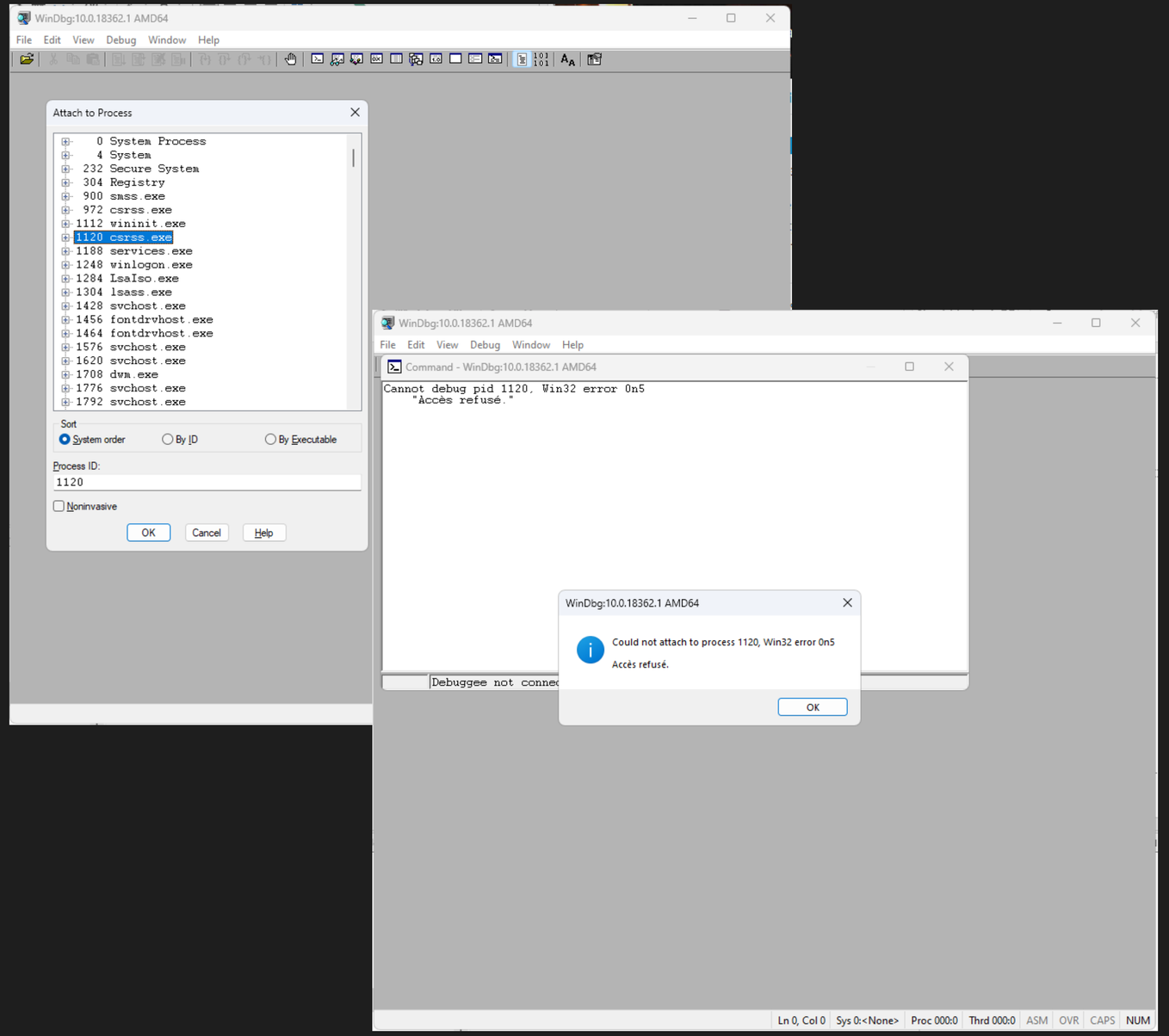
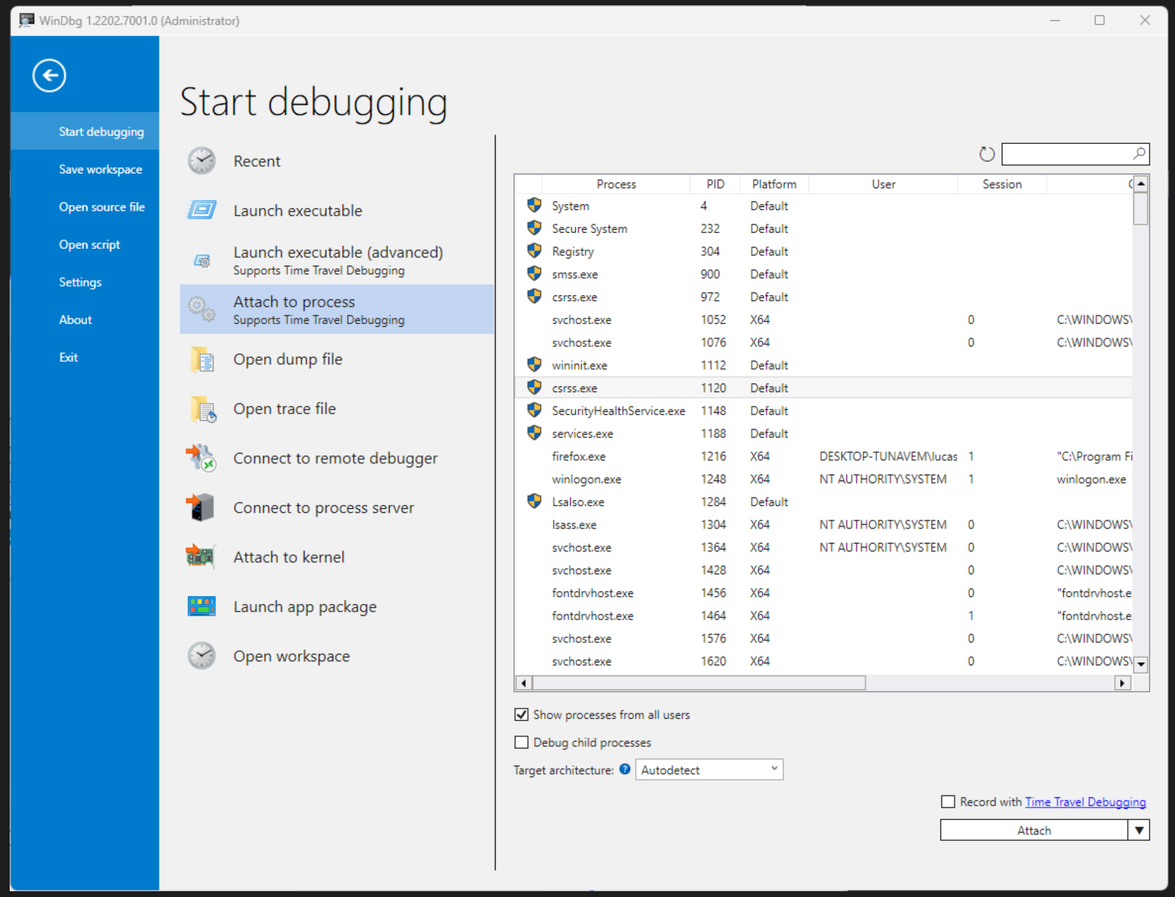
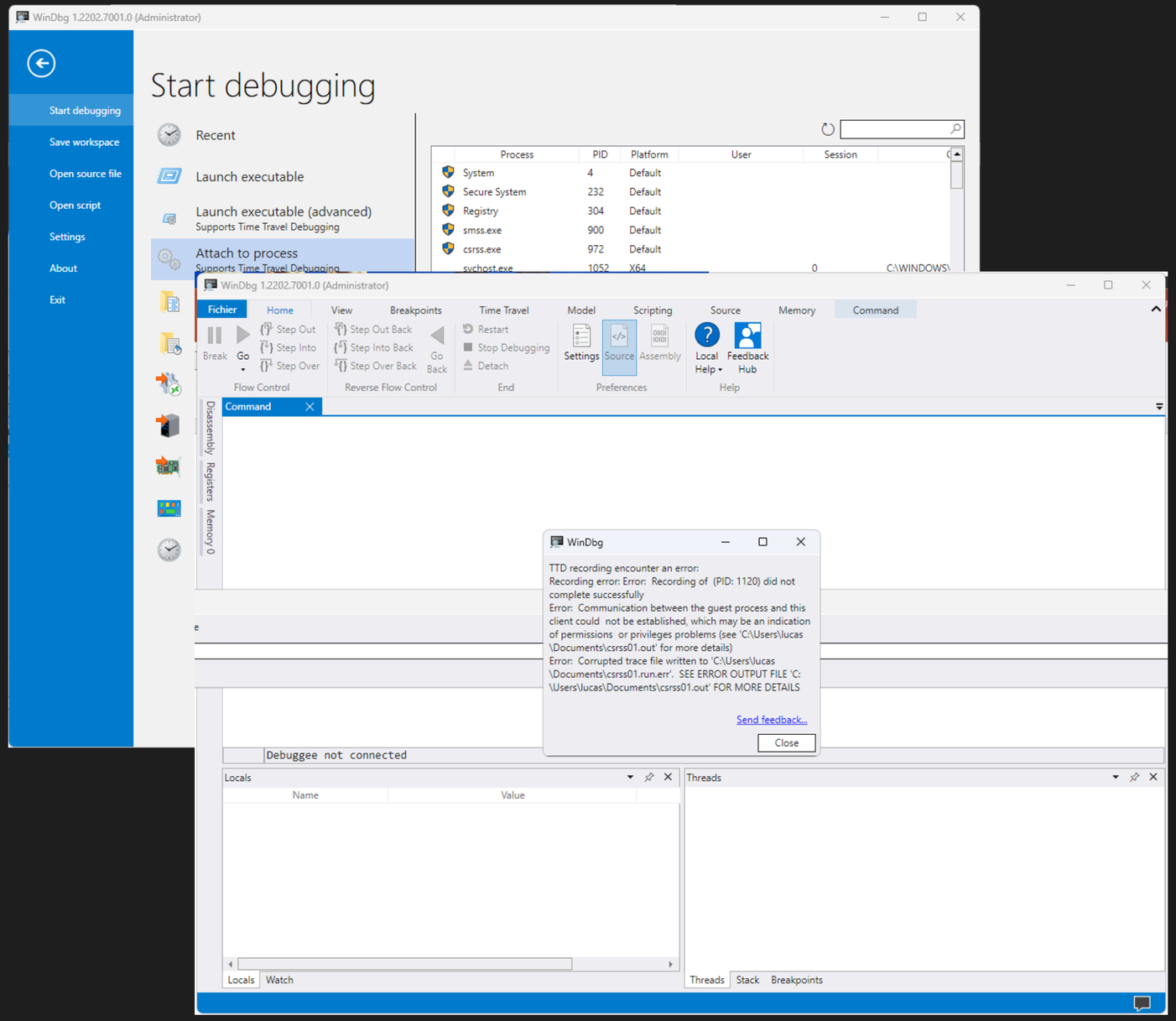
SYNACKTIV

Suprême TTD - That's my PPL


WinDbg:10.0.18362.1 AMD64
File Edit View Debug Window Help
Attach to Process
0 System Process
4 System
232 Secure System
304 Registry
900 smss.exe
972 csrss.exe
1112 wininit.exe
1120 csrss.exe
1188 services.exe
1248 winlogon.exe
1284 LsaIso.exe
1304 lsass.exe
1428 svchost.exe
1456 fontdrvhost.exe
1464 fontdrvhost.exe
1576 svchost.exe
1620 svchost.exe
1708 dwm.exe
1776 svchost.exe
1792 svchost.exe
Sort
System order
By ID
By Executable
Process ID:
1120
Noninvasive
OK
Cancel
Help
Ln 0, Col 0
Sys 0:<None>
Proc 000:0
Thrd 000:0
ASM
OVR
CAPS
NUM


WinDbg:10.0.18362.1 AMD64
File Edit View Debug Window Help
Attach to Process
0 System Process
4 System
232 Secure System
304 Registry
900 smss.exe
972 csrss.exe
1112 wininit.exe
1120 csrss.exe
1188 services.exe
1248 winlogon.exe
1284 LsaIso.exe
1304 lsass.exe
1428 svchost.exe
1456 fontdrvhost.exe
1464 fontdrvhost.exe
1576 svchost.exe
1620 svchost.exe
1708 dwm.exe
1776 svchost.exe
1792 svchost.exe
Sort
System order
By ID
By Executable
Process ID:
1120
Noninvasive
OK
Cancel
Help
WinDbg:10.0.18362.1 AMD64
File Edit View Debug Window Help
Command - WinDbg:10.0.18362.1 AMD64
Cannot debug pid 1120, Win32 error 0n5
"Accès refusé."
Debuggee not connec
WinDbg:10.0.18362.1 AMD64
Could not attach to process 1120, Win32 error 0n5
Accès refusé.
OK
Ln 0, Col 0
Sys 0:<None>
Proc 000:0
Thrd 000:0
ASM
OVR
CAPS
NUM


WinDbg 1.2202.7001.0 (Administrator)
Start debugging
Save workspace
Open source file
Open script
Settings
About
Exit
Start debugging
Recent
Launch executable
Launch executable (advanced)
Supports Time Travel Debugging
Attach to process
Supports Time Travel Debugging
Open dump file
Open trace file
Connect to remote debugger
Connect to process server
Attach to kernel
Launch app package
Open workspace
Process PID Platform User Session
System 4 Default
Secure System 232 Default
Registry 304 Default
smss.exe 900 Default
csrss.exe 972 Default
svchost.exe 1052 X64 0 C:\WINDOWS\
svchost.exe 1076 X64 0 C:\WINDOWS\
wininit.exe 1112 Default
csrss.exe 1120 Default
SecurityHealthService.exe 1148 Default
services.exe 1188 Default
firefox.exe 1216 X64 DESKTOP-TUNAVEM\lucas 1 "C:\Program Fi
winlogon.exe 1248 X64 NT AUTHORITY\SYSTEM 1 winlogon.exe
Lsalso.exe 1284 Default
lsass.exe 1304 X64 NT AUTHORITY\SYSTEM 0 C:\WINDOWS\
svchost.exe 1364 X64 NT AUTHORITY\SYSTEM 0 C:\WINDOWS\
svchost.exe 1428 X64 0 C:\WINDOWS\
fontdrvhost.exe 1456 X64 0 "fontdrvhost.e
fontdrvhost.exe 1464 X64 1 "fontdrvhost.e
svchost.exe 1576 X64 0 C:\WINDOWS\
svchost.exe 1620 X64 0 C:\WINDOWS\
Show processes from all users
Debug child processes
Target architecture:
Autodetect
Record with Time Travel Debugging
Attach


WinDbg 1.2202.7001.0 (Administrator)
Start debugging
Save workspace
Open source file
Open script
Settings
About
Exit
Recent
Launch executable
Launch executable (advanced)
Supports Time Travel Debugging
Attach to process
Supports Time Travel Debugging
Process
PID
Platform
User
Session
System 4 Default
Secure System 232 Default
Registry 304 Default
smss.exe 900 Default
csrss.exe 972 Default
svchost.exe 1052 X64 0 C:\WINDOWS\
WinDbg 1.2202.7001.0 (Administrator)
Fichier
Home
View
Breakpoints
Time Travel
Model
Scripting
Source
Memory
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 Hub
Flow Control
Reverse Flow Control
End
Preferences
Help
Command
Disassembly
Registers
Memory 0
WinDbg
TTD recording encounter an error:
Recording error: Error: Recording of (PID: 1120) did not complete successfully
Error: Communication between the guest process and this client could not be established, which may be an indication of permissions or privileges problems (see 'C:\Users\lucas\Documents\csrss01.out' for more details)
Error: Corrupted trace file written to 'C:\Users\lucas\Documents\csrss01.run.err'. SEE ERROR OUTPUT FILE 'C:\Users\lucas\Documents\csrss01.out' FOR MORE DETAILS
Send feedback...
Close
Debuggee not connected
Locals
Name
Value
Locals
Watch
Threads
Threads
Stack
Breakpoints

Debugging an anti-malware protected service

As part of the protected process security model, other non-protected processes aren't able to inject threads or write into the virtual memory of the protected process. However a kernel debugger (KD) is allowed for debugging any anti-malware protected processes. The KD can also be used to check if the anti-malware service is running as protected or not:

syntax Copy

```
dt -r1 nt!_EPROCESS <Process Address>
+0x67a Protection       : _PS_PROTECTION
      +0x000 Level            : 0x31 '1'
      +0x000 Type             : 0y0001
      +0x000 Signer           : 0y0011
```

If the value of the *Type* member is 0y0001, the service is running as protected.

In addition, only the following SC commands are allowed on anti-malware protected service:

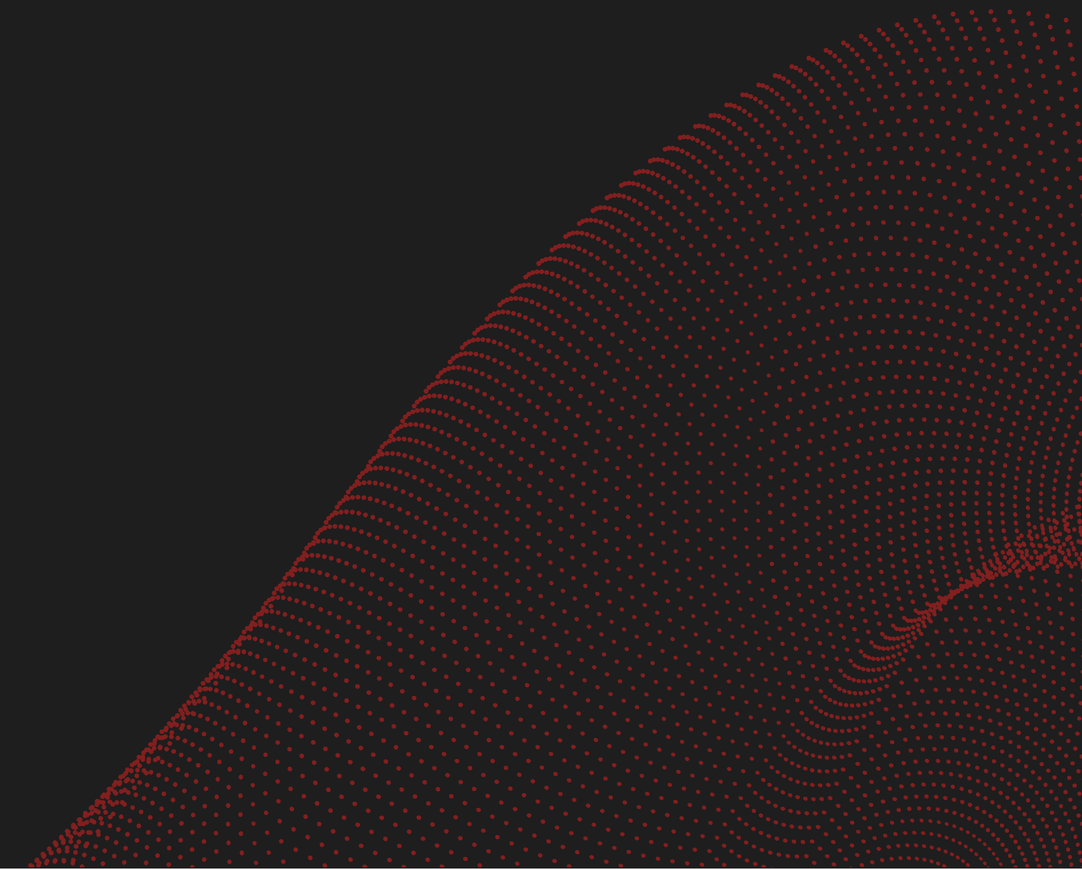
- `sc config start=Auto`
- `sc qc`
- `sc start`
- `sc interrogate`
- `sc sdshow`

If the debugger is attached, use the following flag in the registry to break in the debugger when unsigned (or inappropriately signed) binaries are loaded into the anti-malware protected service.

https://docs.microsoft.com/en-us/windows/win32/services/protecting-anti-malware-services-#debugging-an-anti-malware-protected-service

Thanks for your attention !

Got a question ?



@_lucas_georges_

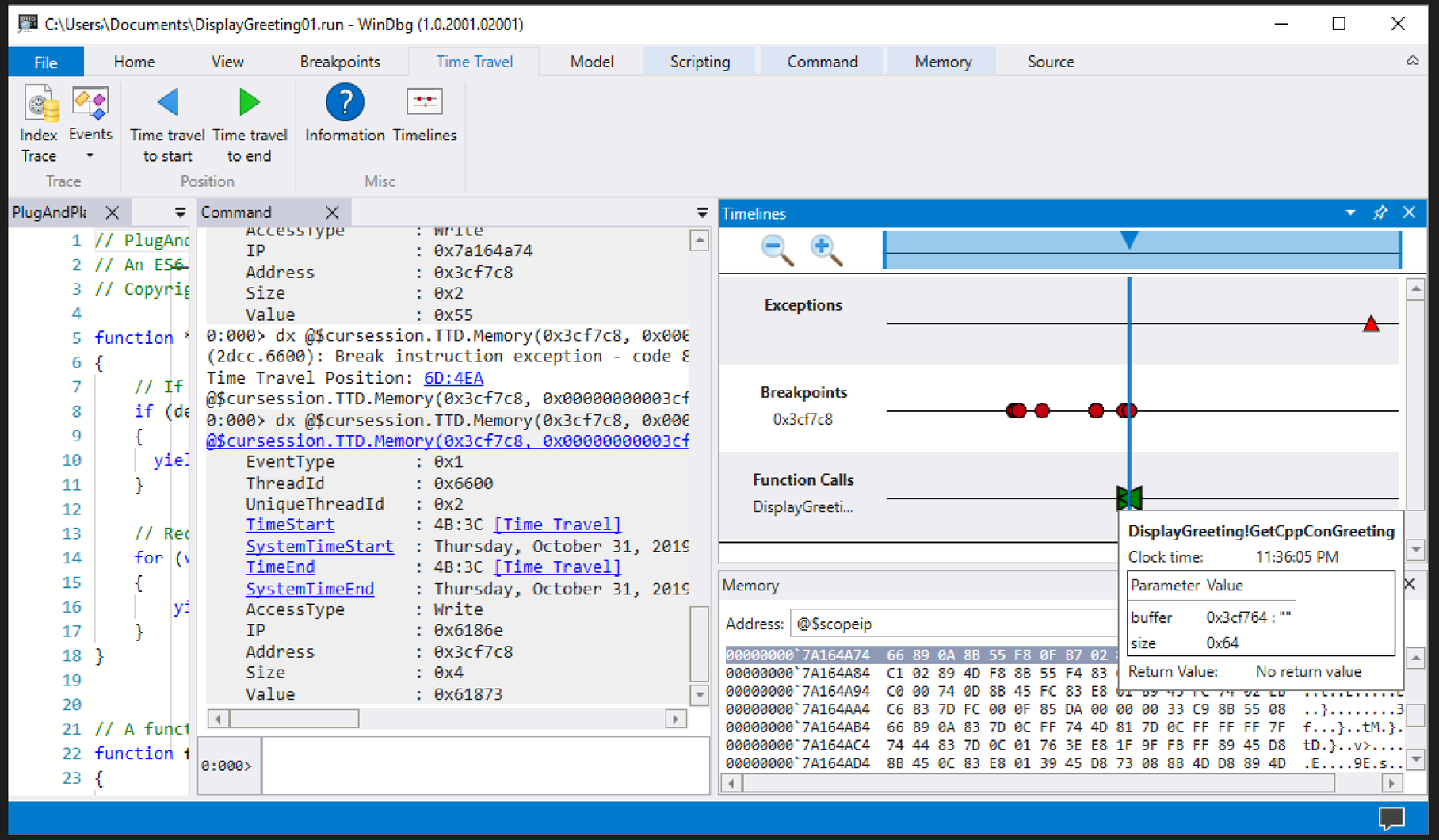
Synacktiv

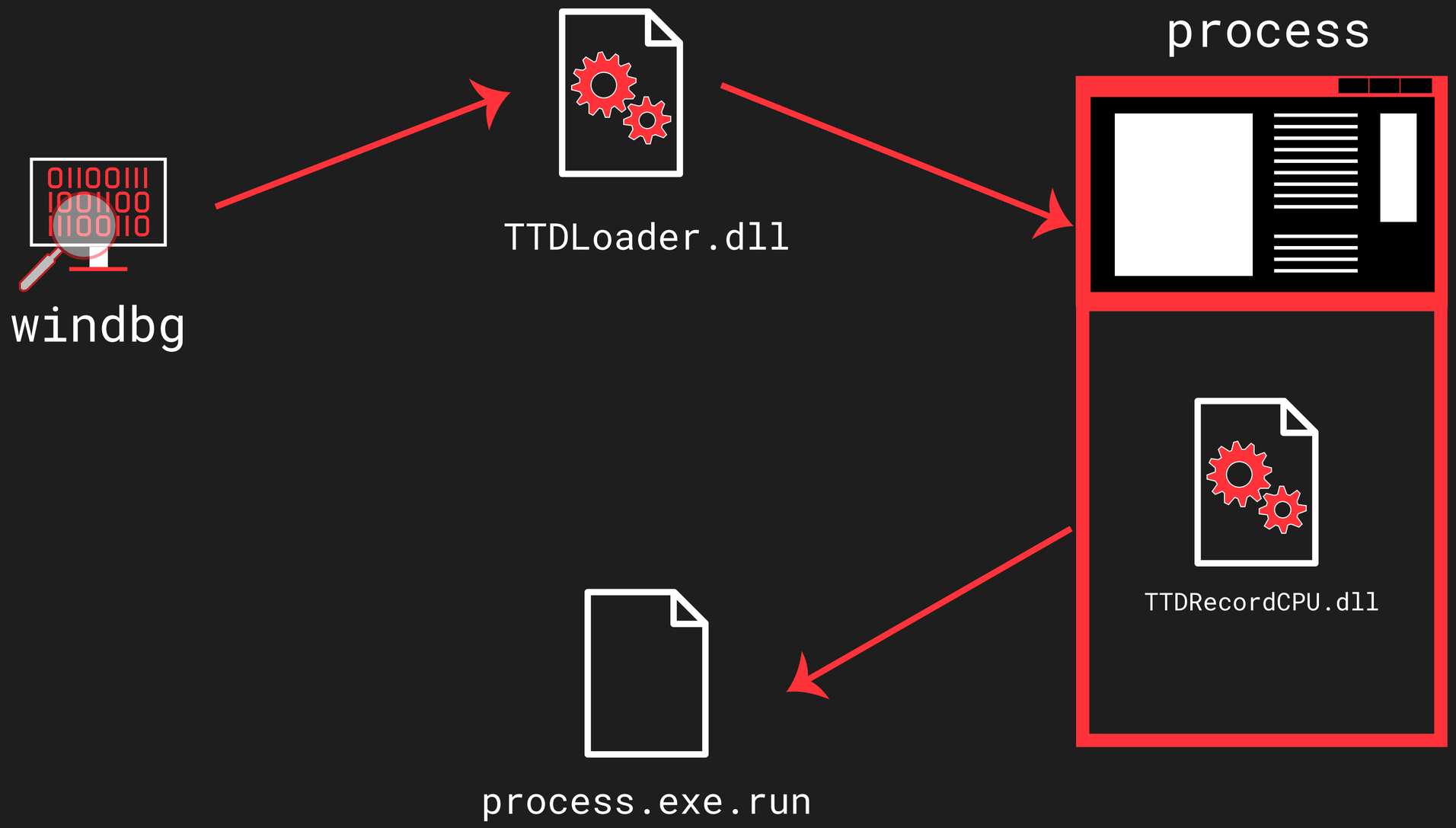
- Offensive security company
- 100+ ninjas
- 4 departments
  - Pentest
  - Reverse Engineering
  - Development
  - Incident Response

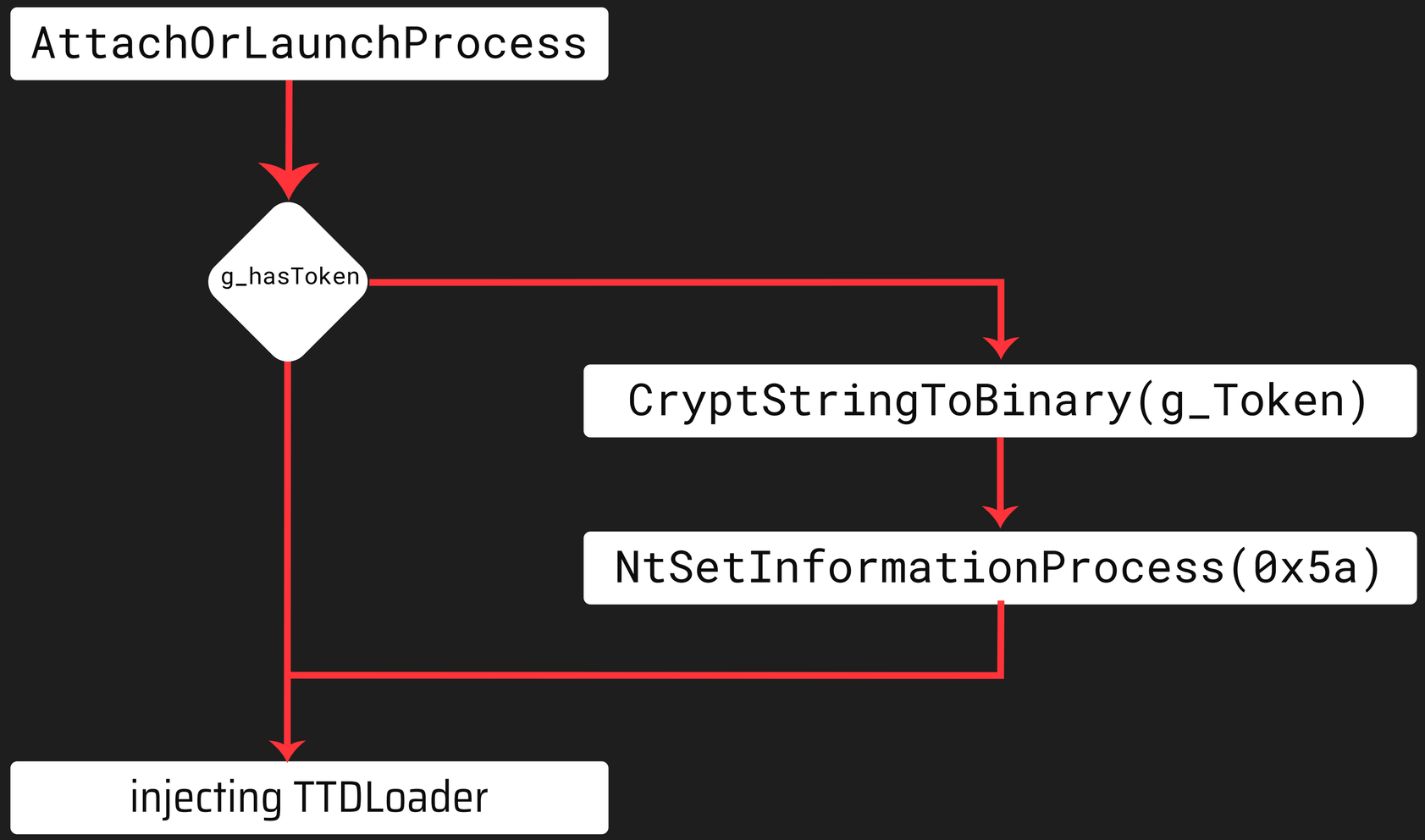
Time Travel Debugging

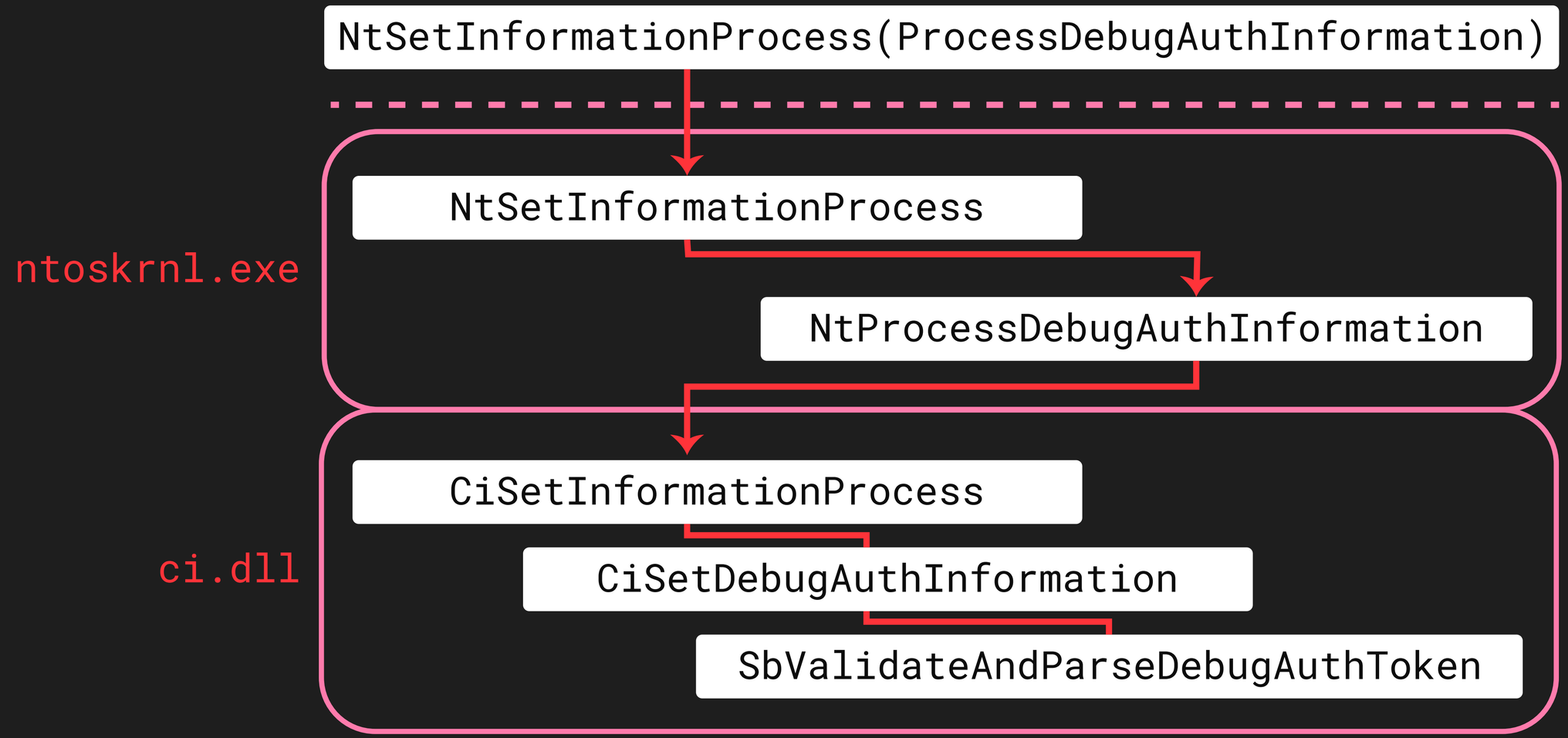
Time travel debugging

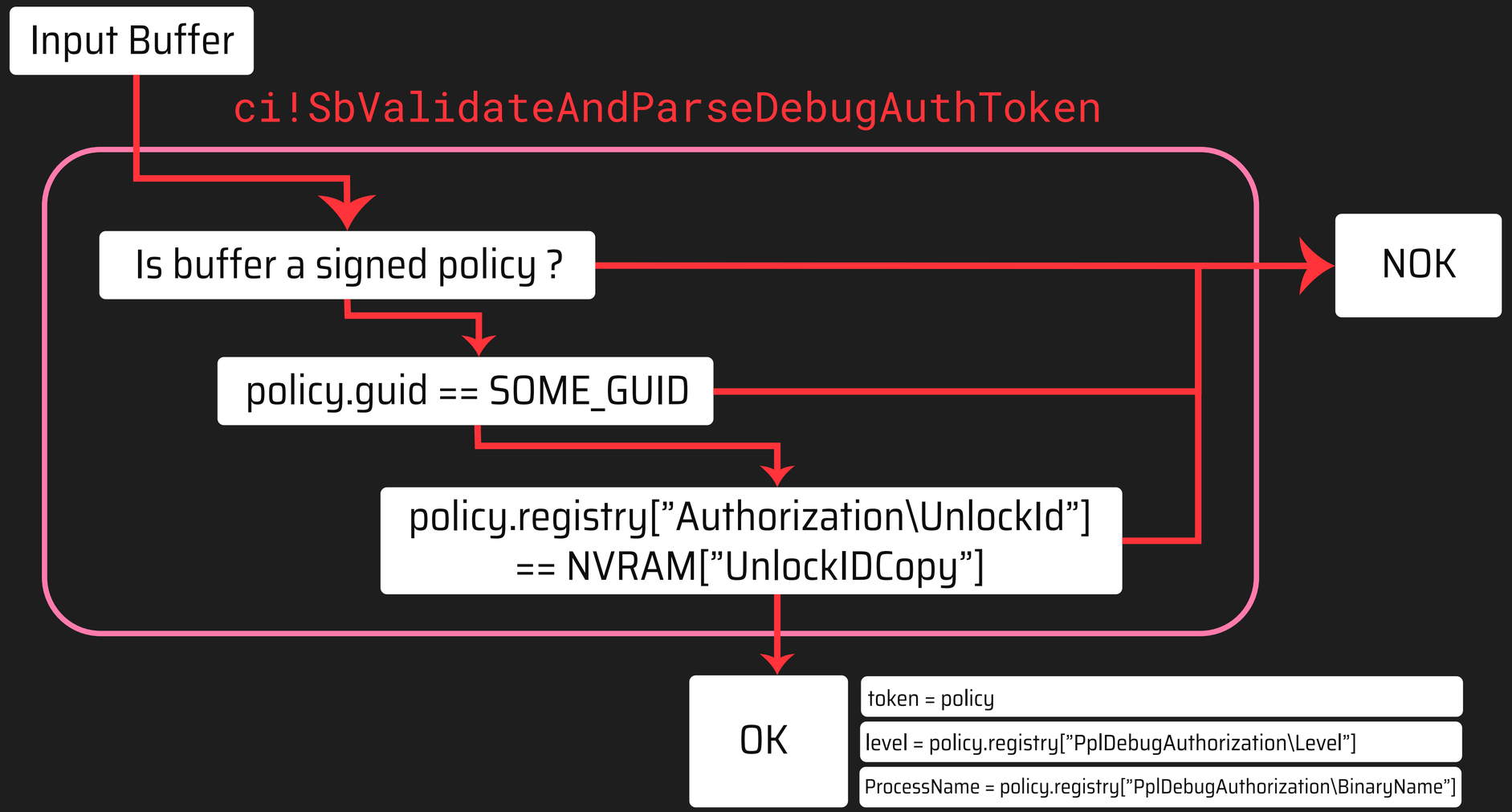
- rr : https://rr-project.org/
- Qira : http://qira.me/
- iDNA/Nirvana/TTD

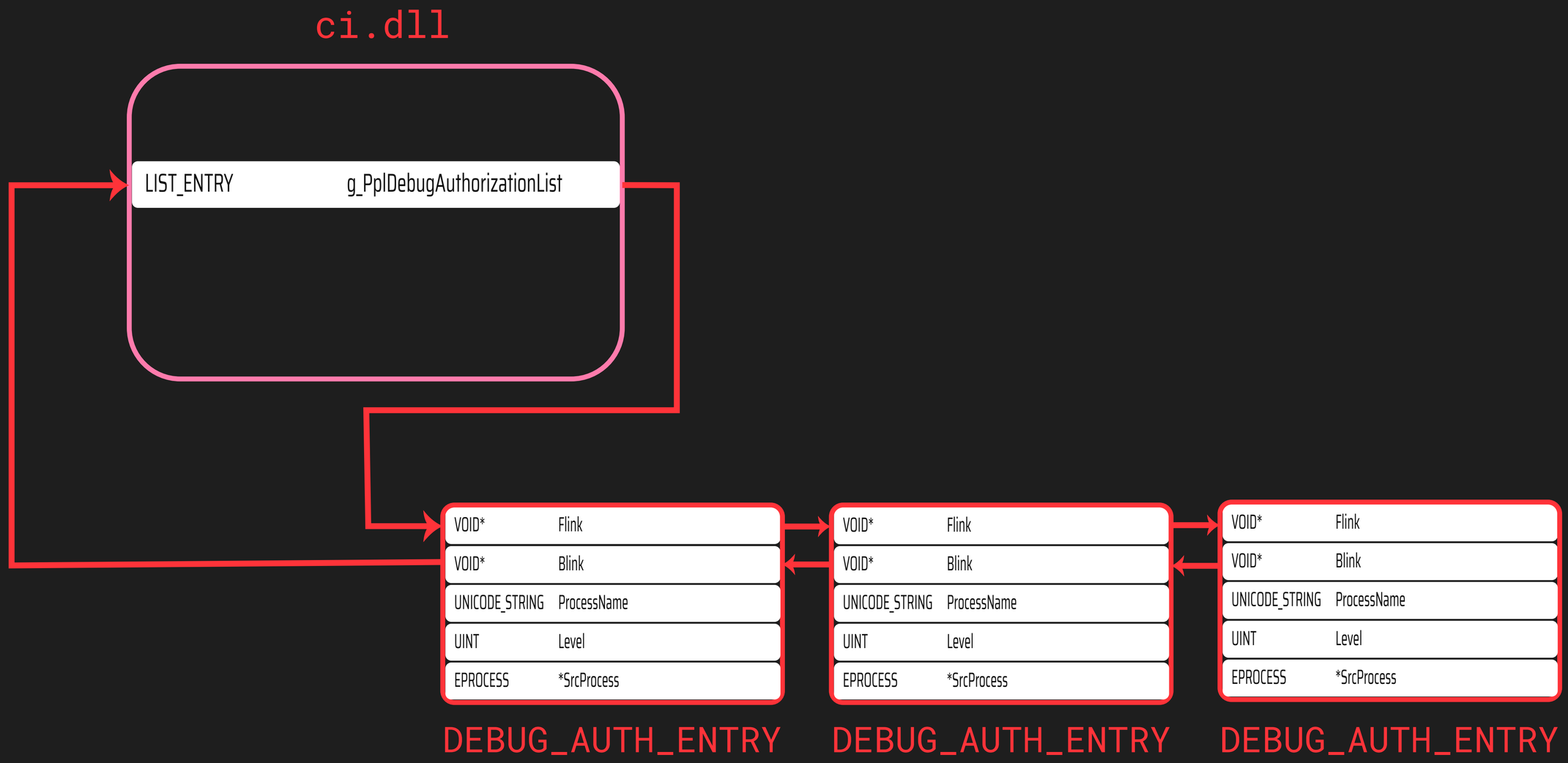

C:\Users\Documents\DisplayGreeting01.run - WinDbg (1.0.2001.02001)
File
Home
View
Breakpoints
Time Travel
Model
Scripting
Command
Memory
Source
Index Trace
Events
Time travel to start
Time travel to end
Information
Timelines
Trace
Position
Misc
PlugAndPl
Command
Timelines
Exceptions
Breakpoints
0x3cf7c8
Function Calls
DisplayGreeti...
DisplayGreeting!GetCppConGreeting
Clock time: 11:36:05 PM
Parameter Value
buffer 0x3cf764 : ""
size 0x64
Return Value: No return value
Memory
Address: @$scopeip
IP : 0x7a164a74
Address : 0x3cf7c8
Size : 0x2
Value : 0x55
0:000> dx @$cursession.TTD.Memory(0x3cf7c8, 0x000
(2dcc.6600): Break instruction exception - code 8
Time Travel Position: 6D:4EA
@$cursession.TTD.Memory(0x3cf7c8, 0x00000000003cf
0:000> dx @$cursession.TTD.Memory(0x3cf7c8, 0x000
@$cursession.TTD.Memory(0x3cf7c8, 0x00000000003cf
EventType : 0x1
ThreadId : 0x6600
UniqueThreadId : 0x2
TimeStart : 4B:3C [Time Travel]
SystemTimeStart : Thursday, October 31, 2019
TimeEnd : 4B:3C [Time Travel]
SystemTimeEnd : Thursday, October 31, 2019
AccessType : Write
IP : 0x6186e
Address : 0x3cf7c8
Size : 0x4
Value : 0x61873
0:000>
00000000`7A164A74 66 89 0A 8B 55 F8 0F B7 02
00000000`7A164A84 C1 02 89 4D F8 8B 55 F4 83
00000000`7A164A94 C0 00 74 0D 8B 45 FC 83 E8
00000000`7A164AA4 C6 83 7D FC 00 0F 85 DA 00 00 00 33 C9 8B 55 08
00000000`7A164AB4 66 89 0A 83 7D 0C FF 74 4D 81 7D 0C FF FF FF 7F
00000000`7A164AC4 74 44 83 7D 0C 01 76 3E E8 1F 9F FB FF 89 45 D8
00000000`7A164AD4 8B 45 0C 83 E8 01 39 45 D8 73 08 8B 4D D8 89 4D


process
TTDLoader.dll
windbg
TTDRecordCPU.dll
process.exe.run


AttachOrLaunchProcess
g_hasToken
CryptStringToBinary(g_Token)
NtSetInformationProcess(0x5a)
injecting TTDLoader


NtSetInformationProcess(ProcessDebugAuthInformation)
ntoskrnl.exe
NtSetInformationProcess
NtProcessDebugAuthInformation
ci.dll
CiSetInformationProcess
CiSetDebugAuthInformation
SbValidateAndParseDebugAuthToken


Input Buffer
ci!SbValidateAndParseDebugAuthToken
Is buffer a signed policy ?
NOK
policy.guid == SOME_GUID
policy.registry[”Authorization\UnlockId”]
== NVRAM[”UnlockIDCopy”]
OK
token = policy
level = policy.registry[”PplDebugAuthorization\Level”]
ProcessName = policy.registry[”PplDebugAuthorization\BinaryName”]


ci.dll
LIST_ENTRY g_PplDebugAuthorizationList
VOID* Flink
VOID* Blink
UNICODE_STRING ProcessName
UINT Level
EPROCESS *SrcProcess
DEBUG_AUTH_ENTRY
VOID* Flink
VOID* Blink
UNICODE_STRING ProcessName
UINT Level
EPROCESS *SrcProcess
DEBUG_AUTH_ENTRY
VOID* Flink
VOID* Blink
UNICODE_STRING ProcessName
UINT Level
EPROCESS *SrcProcess
DEBUG_AUTH_ENTRY

DEMO

Conclusion

Conclusion

- PplDebugAuthorization
  - Introduced around windows 8
  - Solid cryptographic code
- No runtime attestation
  - No telemetry events
  - Limited check from PatchGuard
- POC is left as exercice for the reader

Thanks for your attention !

Got a question ?