MORPHISEC

Breach Prevention Made Easy

Enhanced CTI with runtime memory forensics

Michael Gorelik CTO, Morphisec, CISSP

aws partner network

Select Technology Partner

Member of Microsoft Intelligent Security Association

Microsoft

ivanti ONE CERTIFIED

Available in the Microsoft Azure Marketplace

PARTNER READY VMWARE HORIZON

CITRIX Ready

U.S. Department of Homeland Security

CYBER THREAT ALLIANCE

About Michael

Michael Gorelik

Chief Technology Officer
Morphisec

Fast Facts

- Co-Founder, CTO and Head of Malware Research at Morphisec
- Noted speaker, having presented at multiple industry conferences, such SANS, BSides, and RSA
- Jointly holds seven patents in the IT space

Methodology

- Morphisec is an endpoint solution focusing on memory & runtime protection.
- Installed in over 5,000 organizations, 9m+ endpoints.
- Alert logs are collected from customers and analyzed by our Forensic team.

CTI with runtime memory forensics

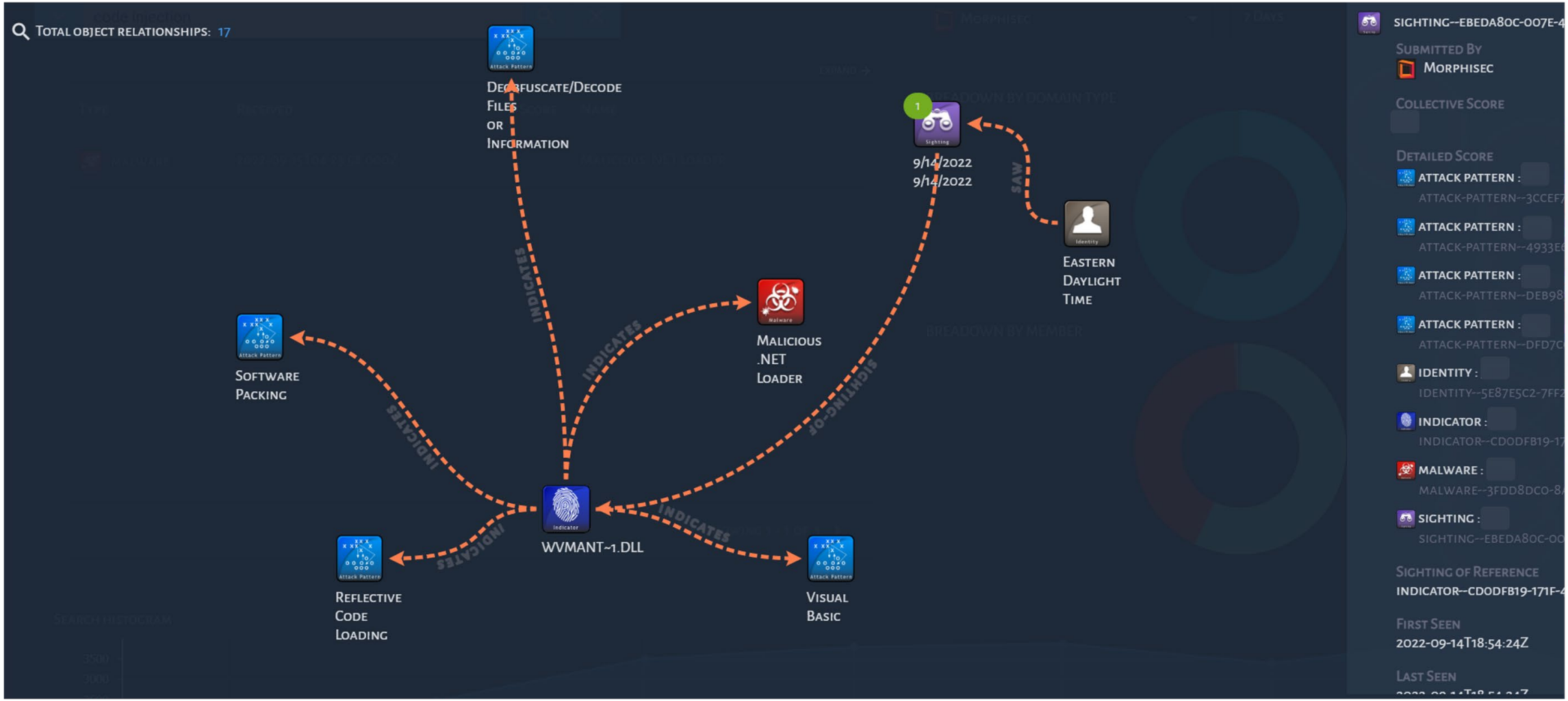
"Cyber threat intelligence is all about knowing what your adversaries do and using that information to improve decision-making."

https://medium.com/mitre-attack/getting-started-with-attack-cti-4eb205be4b2f

"Utilizing memory forensics during incident response provides valuable cyber threat intelligence. By both providing mechanisms to verify current compromise using known indicators and to discover additional indicators, memory forensics can be leveraged to identify, track, isolate and remediate more efficiently. "

https://www.sans.org/white-papers/34162/

Representative CTI STIX bundle example



Fileless In-memory trend

Fileless malware has seen dramatic increase over the past year, why?

- Things that are not landing on disk are much harder to detect
- Threat usually do not leave significant forensic evidence

Can CTI provide good enough coverage?

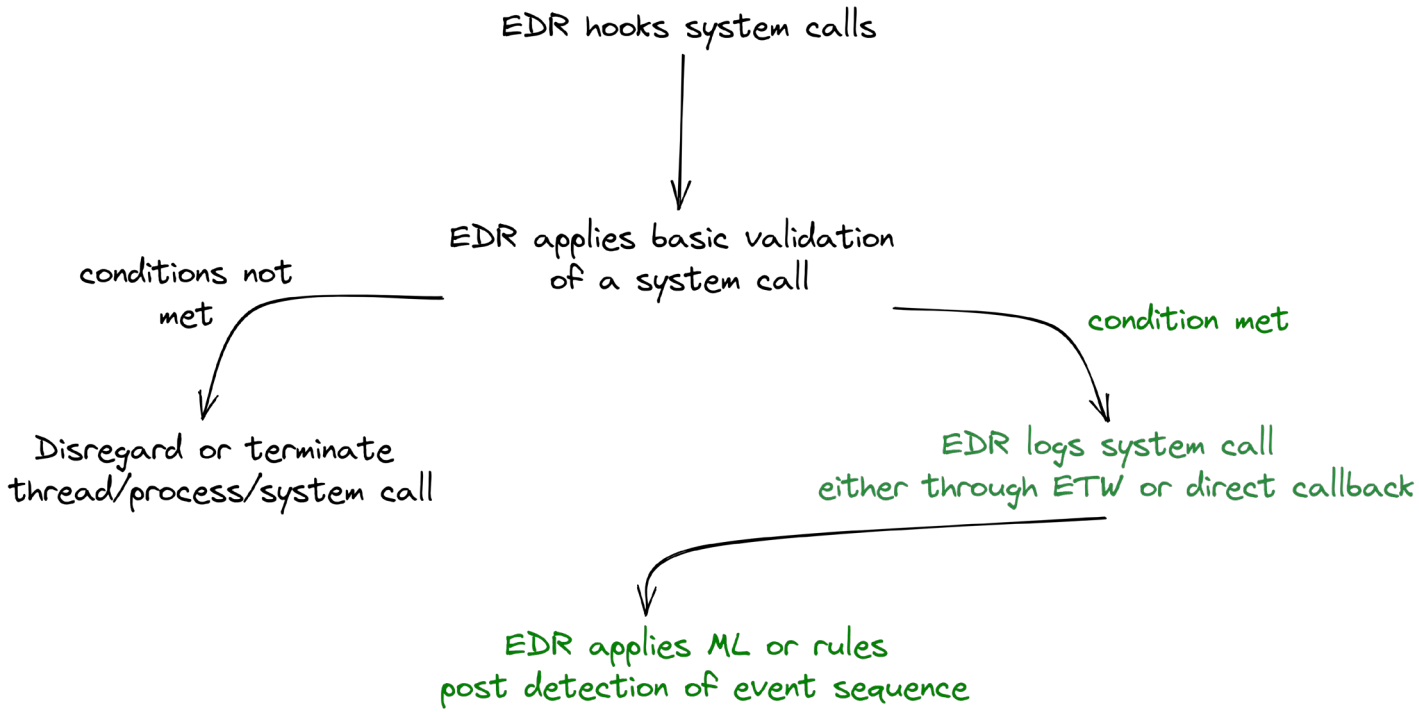
Fileless attack categories:

- Exploits – hijacking the flow of existing application
- Interpreter scripts – PowerShell, VBScript, JavaScript,…
- Code injections – executing implants or known tools from memory
- Lolbins – allowlisting bypass through proxy execution

Standard Runtime Detection Flow

Example of validation logic:

- System call executed on remote process or thread (remote injection)
- During the execution there is violation of stack (exploit)
- System call return address is outside of defined region (exploit or shellcode)
- System call parameters may indicate future shellcode execution (page execute)



Detecting runtime patterns Using memory scanners

Due to fileless nature of threats – memory scanners become popular. More and more EDRs now started to monitor memory for potentially malicious code implants

Known drawbacks of memory scanners:

- **Avoiding alertable states (Sleep)**
- **Encryption and obfuscation of pages and data**
- **Targeting minimal IOCs**
- **Impact on usability due to large scan surface**
- Return address spoofing
- Targeting mostly executable private commit memory
- ROP execution

Popular memory scanners:

- Moneta
- Pe-sieve
- BeaconHunter
- Patriot
- BeaconEye
- MalMemDetect
- Volatility-Malfind

Highly recommend to review Kyle's Avery presentation on avoiding memory scanners (Defcon 30th)

EDR extensions for process runtime observables data (STIX 2.1)

Specifications

Property Name	Type	Description
type (required)	string	The value of this property **MUST** be process.
extensions (optional)	dictionary	The Process object defines the following extensions. In addition to these, producers **MAY** create their own. windows-process-ext, windows-service-ext Dictionary keys **MUST** identify the extension type by name. The corresponding dictionary values **MUST** contain the contents of the extension instance.
is_hidden (optional)	boolean	Specifies whether the process is hidden.
pid (optional)	integer	Specifies the Process ID, or PID, of the process.
created_time (optional)	timestamp	Specifies the date/time at which the process was created.
cwd (optional)	string	Specifies the current working directory of the process.

Paloalto

process	extensions.x-paloalto-process.termination_code	action_process_termination_code
process	extensions.x-paloalto-process.termination_date	action_process_termination_date
process	extensions.x-paloalto-process.tid	action_remote_process_thread_id
process	extensions.x-paloalto-process.instance_exec_time	action_process_instance_execution_time
process	extensions.x-paloalto-process.execution_time	actor_process_execution_time

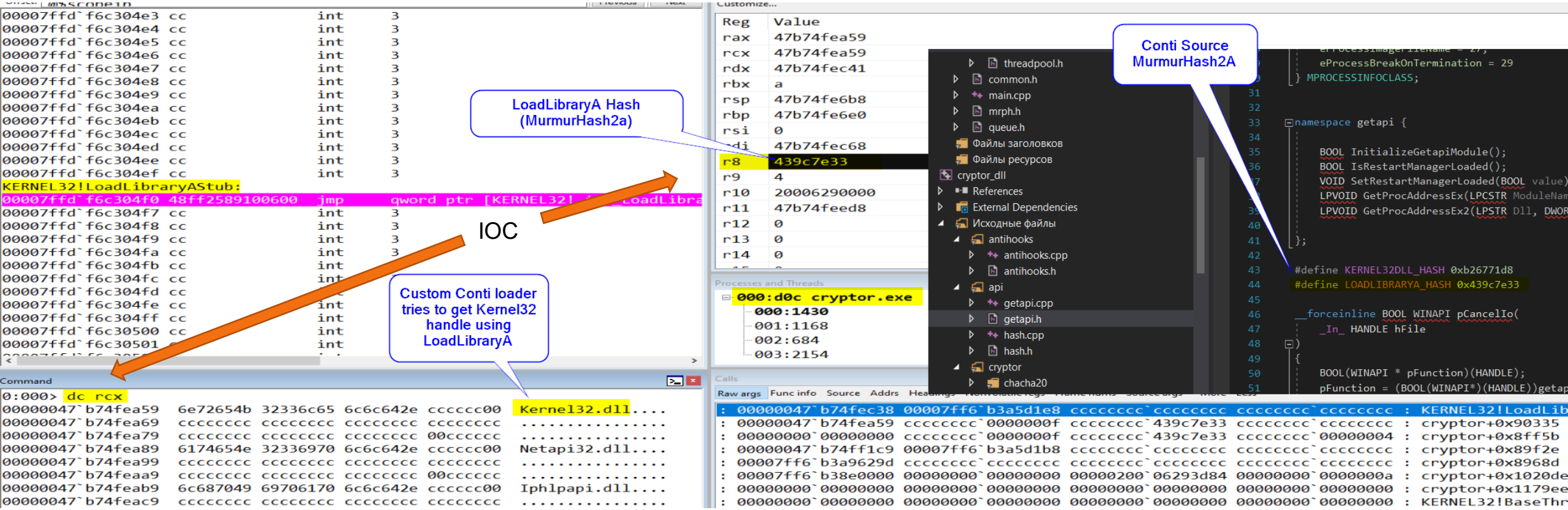
SentinelOne

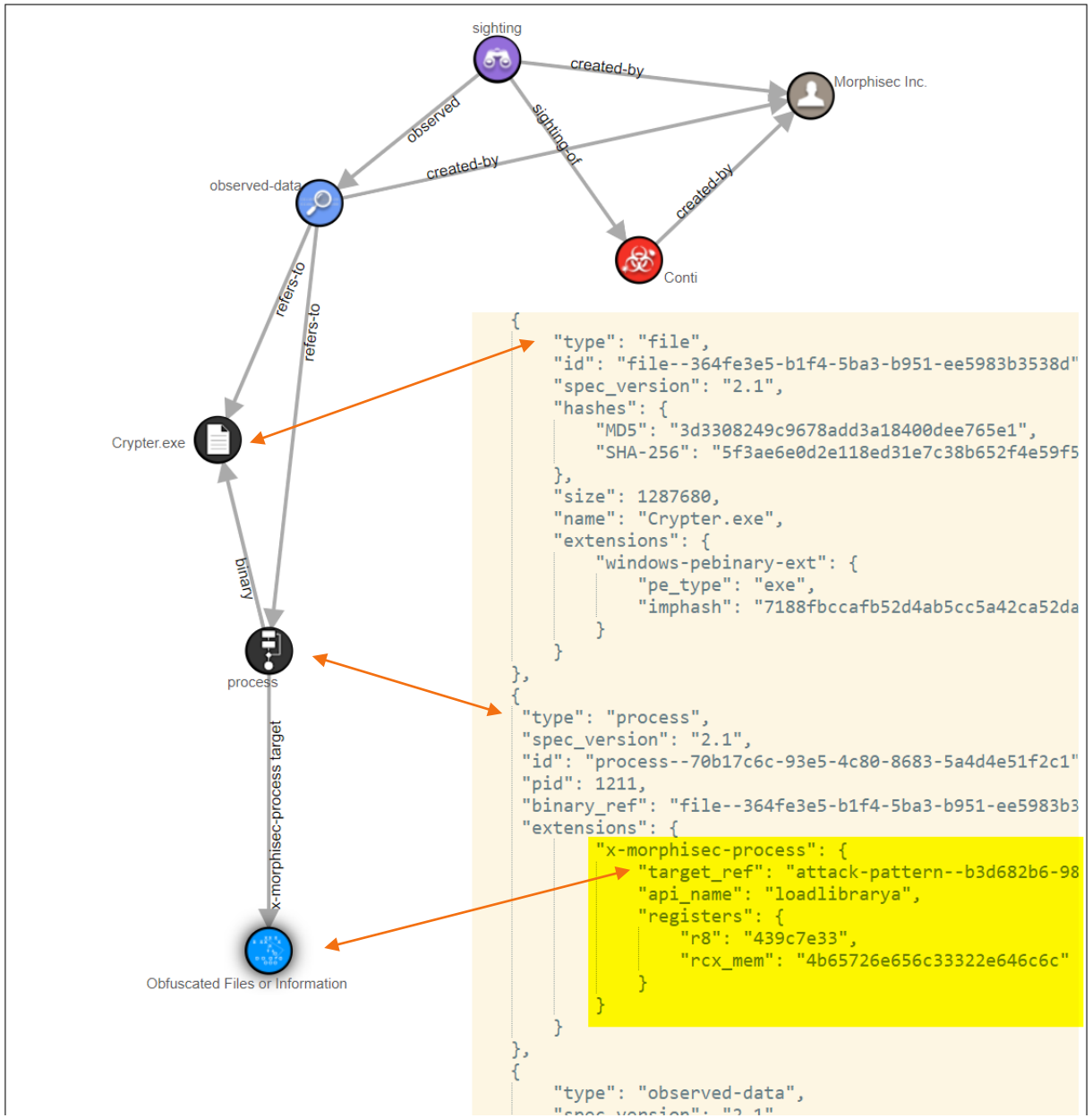
process	extensions.x-sentinelone-process.story_line_id	srcProcStorylineId
process	extensions.x-sentinelone-process.story_line_id	tgtProcStorylineId
process	extensions.x-sentinelone-process.integrity_level	srcProcIntegrityLevel
process	extensions.x-sentinelone-process.integrity_level	tgtProcIntegrityLevel
process	extensions.x-sentinelone-process.process_unique_id	srcProcUid

Cybereason

process	extensions.x-cybereason-process.modules_from_temp	modulesFromTemp
process	extensions.x-cybereason-process.unsigned_signed_version_module	unsignedWithSignedVersionModules
process	extensions.x-cybereason-process.unwanted_classification_modules	unwantedClassificationModules
process	extensions.x-cybereason-process.external_connection_evidence	hasRareExternalConnectionEvidence
process	extensions.x-cybereason-process.remote_address_evidence	hasRareRemoteAddressEvidence

Runtime detection of Conti ransomware





Legend

- Identity
- Malware
- File
- Process
- Observed-data
- Attack-pattern
- Sighting

Runtime detection of Conti ransomware

<u>Malicious process behavior:</u>

- When process tries try to load kernel32 &
- R8 register points to the hash value of “LoadLibraryA” (MurmurHash2A)
- This behavior mapped to attack pattern technique “Obfuscated Files or Information (T1027)”

Runtime detection of Metasploit MSFvenom (6.1.37-dev) implant

```
No prior disassembly possible
KERNELBASE!VirtualAlloc:
00007ff8`3dff6ec0 4053            push    rbx
00007ff8`3dff6ec2 4883ec30        sub     rsp,30h
00007ff8`3dff6ec6 33db            xor     ebx,ebx
00007ff8`3dff6ec8 4889542448      mov     qword ptr [rsp+48h],rdx
00007ff8`3dff6ecd 48894c2440      mov     qword ptr [rsp+40h],rcx
00007ff8`3dff6ed2 4885c9          test    rcx,rcx
00007ff8`3dff6ed5 740f            je      KERNELBASE!VirtualAlloc+0x26 (00007ff8`3dff6ee6)
00007ff8`3dff6ed7 8b05bb902d00    mov     eax,dword ptr [KERNELBASE!SysInfo+0x18 (00007ff8`3e2cff98)]
00007ff8`3dff6edd 483bc8          cmp     rcx,rax
```

```
R.. Value
rax 0
rcx 0
rdx 3d000
rbx 2659022ab75
rsp bf2c50ce18
rbp 0
rsi 2659022ac75
rdi 7ff8407c5900
r8  103000
```

Natural parameters of VirtualAlloc

Not a parameter, but points to the implant

Command

```
1:042> dc rbx
00000265`9022ab75  52415a4d e5894855 20ec8348 f0e48348  MZARUH..H.. H...
00000265`9022ab85  00[illegible]e8 81485b00 005af3c3 48d3ff00  .....[H...Z....H
00000265`9022ab95  06f8c38[illegible]9490003 5a046ad8 0000d0ff  ......I..j.Z....
00000265`9022aba5  00000000 0[illegible]00 00000000 00000100  ................
00000265`9022abb5  0eba1f0e cd09b4[illegible]01b821 685421cd  ........!..L.!Th
00000265`9022abc5  70207369 72676f72 6[illegible]6f6e6e61  is program canno
00000265`9022abd5  65622074 6e757220 206e6[illegible]4f44  t be run in DOS
00000265`9022abe5  65646f6d 0a0d0d2e 00000024 [illegible]  mode....$.......
1:042> !vprot rbx
BaseAddress:       000002659022a000
AllocationBase:    0000026580000000
AllocationProtect: 00000004  PAGE_READWRITE
RegionSize:        0000000000038000
State:             00001000  MEM_COMMIT
Protect:           00000040  PAGE_EXECUTE_READWRITE
Type:              00020000  MEM_PRIVATE
1:042> da (rbx+5B08)
00000265`9023067d  "@SUVWATAUAVAWH..83.D..H..$."
```

```
1:042> u rbx
00000265`9022ab75 4d5a          pop     r10
00000265`9022ab77 4152          push    r10
00000265`9022ab79 55            push    rbp
00000265`9022ab7a 4889e5        mov     rbp,rsp
00000265`9022ab7d 4883ec20      sub     rsp,20h
00000265`9022ab81 4883e4f0      and     rsp,0FFFFFFFFFFFFFFF0h
00000265`9022ab85 e800000000    call    00000265`9022ab8a
00000265`9022ab8a 5b            pop     rbx
```

IOC

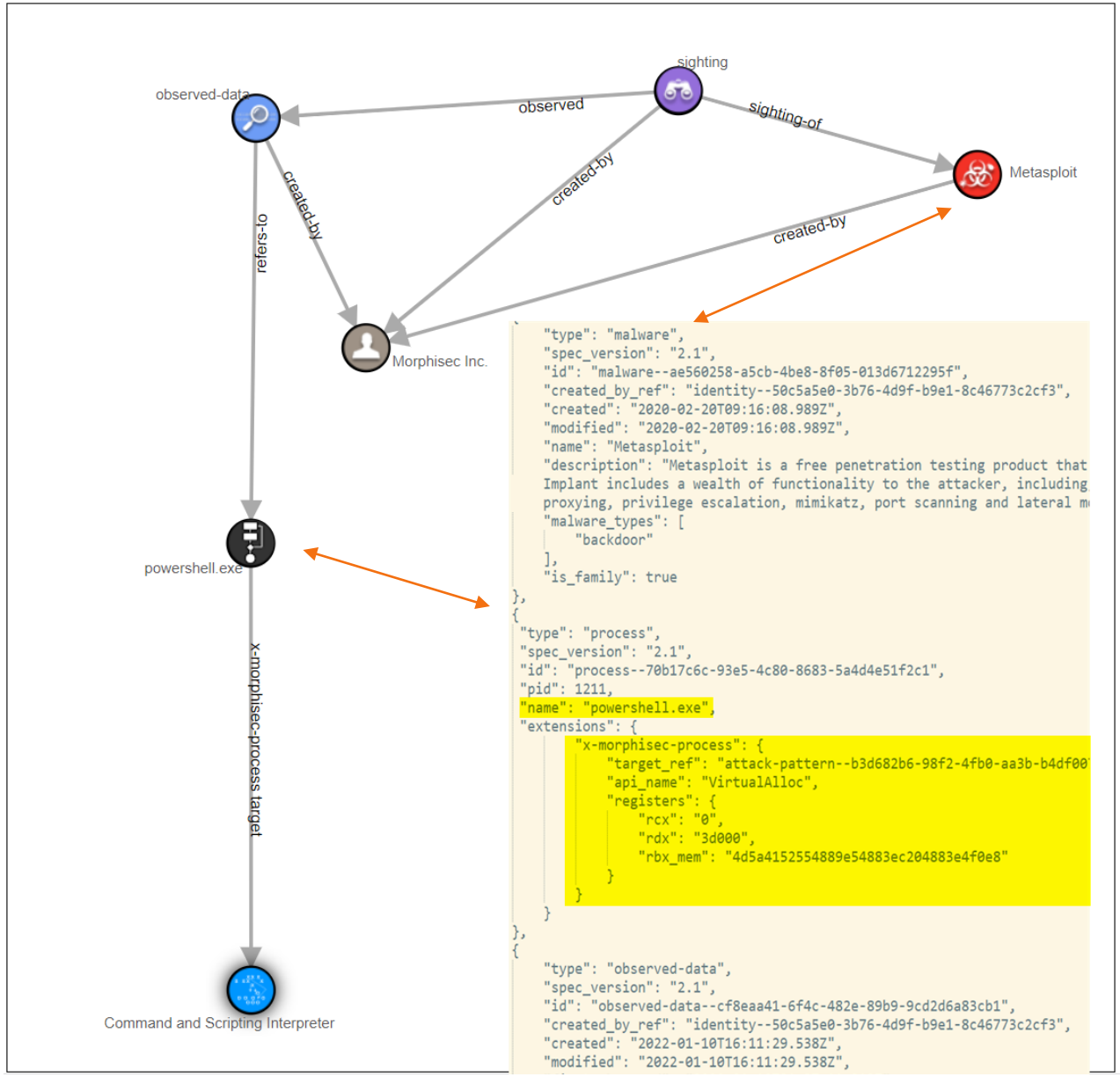
Processes and Threads

```
000:61bc wscript.exe
001:1e5c powershell.exe
002:63d8 conhost.exe
```

Calls

Raw args | Func info | Source | Addrs | Headings | Nonvolatile regs | Frame nums | Source args | More | Less

```
000000bf`2c50ce18 00000265`90230954 KERNELBASE!VirtualAlloc
000000bf`2c50ce20 00000000`00036a8d 0x00000265`90230954
000000bf`2c50ce28 00000265`80003870 0x36a8d
000000bf`2c50ce30 000000bf`2c50ce38 0x00000265`80003870
000000bf`2c50ce38 00009748`ccaa89b5 0x000000bf`2c50ce38
000000bf`2c50ce40 00000265`e7db52e0 0x00009748`ccaa89b5
000000bf`2c50ce48 00000265`9022ac75 0x00000265`e7db52e0
000000bf`2c50ce50 00007ff7`d12410a8 0x00000265`9022ac75
000000bf`2c50ce58 00000000`00000000 0x00007ff7`d12410a8
```



Legend

- Identity
- Malware
- Process
- Observed-data
- Attack-pattern
- Sighting

Runtime detection of Metasploit implant

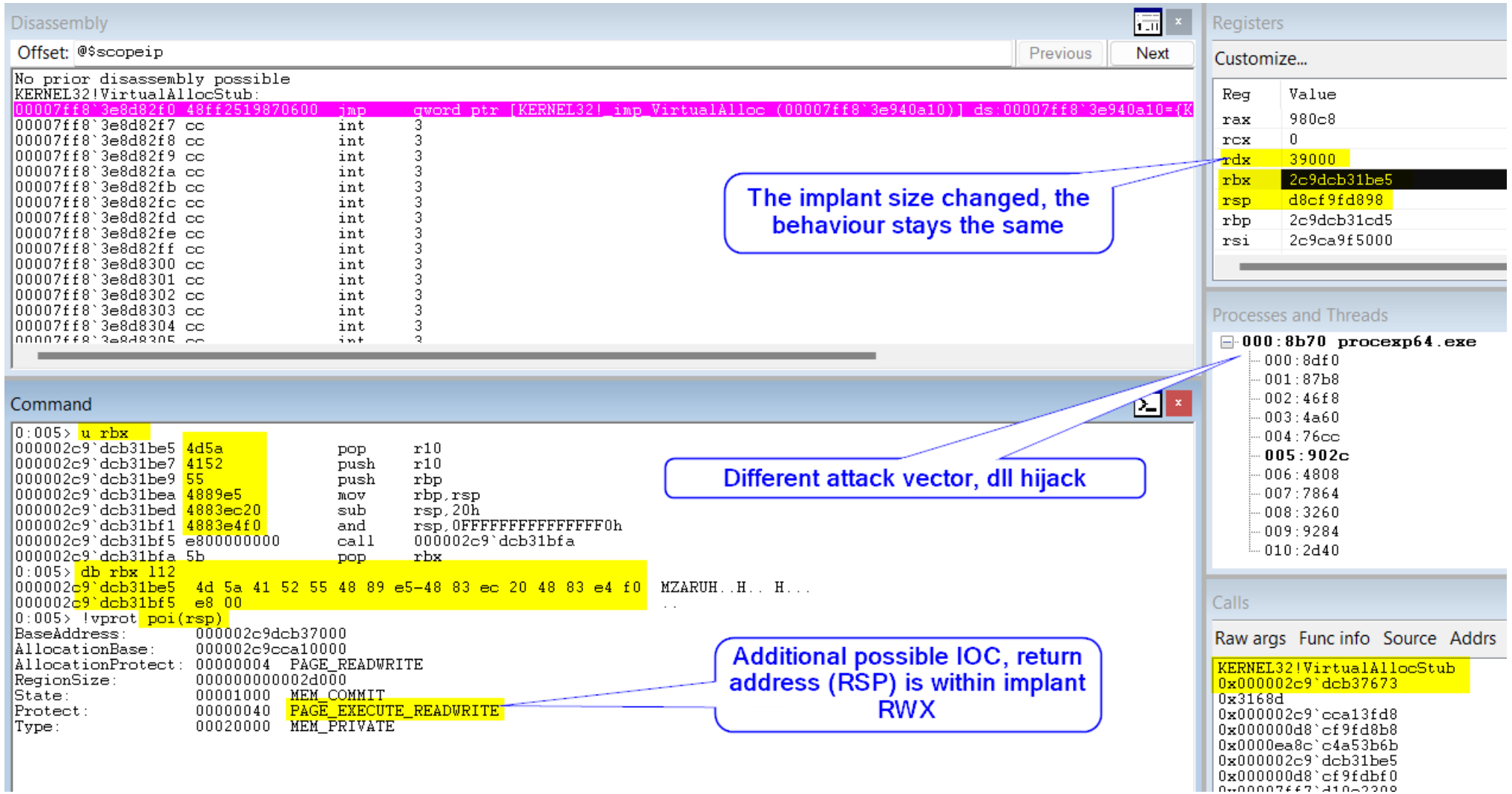
Challenges to present Metasploit / Cobalt?

- Is this malware or indicator? Do we have known URL or Hash or maybe only process behavior (observable-data)?

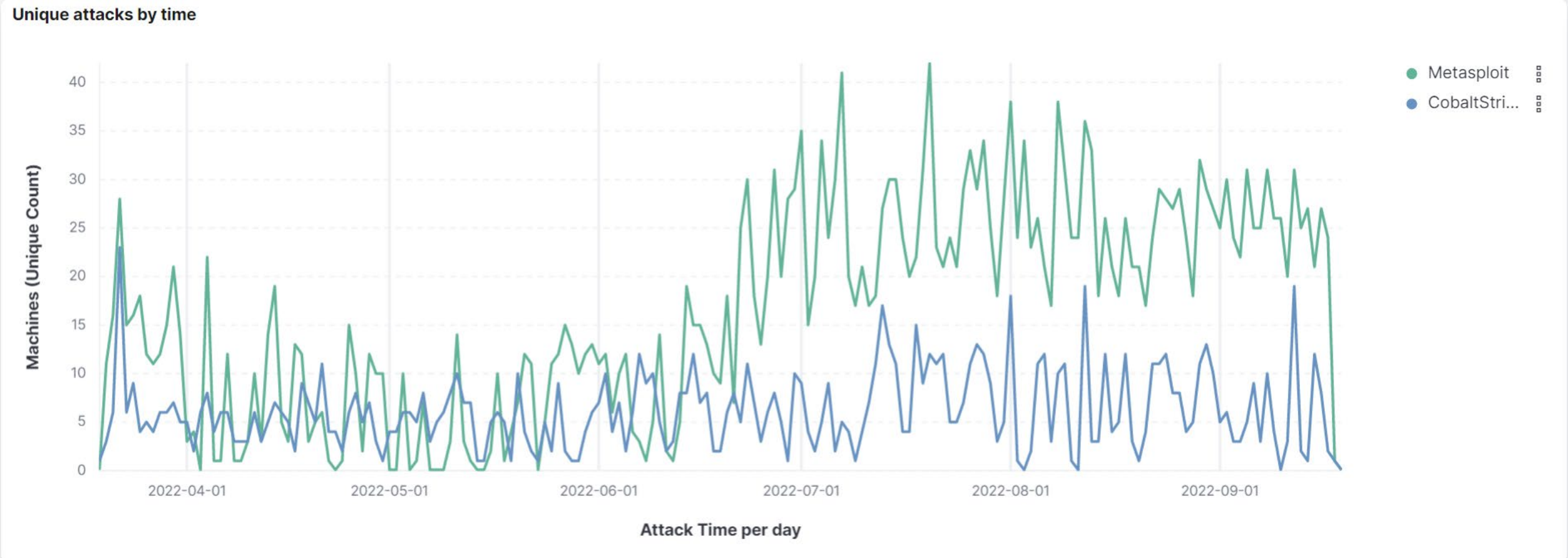
Metasploit process example behavior:

- Allocating memory for the implant (msfvenom) – size 3d00 (address=0)
- One of the registers points to the implant within a temp memory

Runtime detection of Metasploit MSFvenom (5.0.88-dev) implant



Some stats for cobalt-strike



How to generate runtime CTI

- Manual analysis + combination of offline memory scanners
- Enhanced visibility reporting feed (filtered ETW + memory visibility)
- Prevention + deception technology (Moving Target Defense)

Final Notes

- Attack-patterns representing process behavior state have to be standardized, STIX 2.1 facilitates the change.
- Techniques have been presented on how to improve detection based on process state without the need for memory scanning
- Infrastructure already exists to generate and utilize runtime CTI, this will significantly slower down the threat actors

References

- https://oasis-open.github.io/cti-documentation/stix/intro
- https://media.defcon.org/DEF%20CON%2030/DEF%20CON%2030%20presentations/Kyle%20Avery%20-%20Avoiding%20Memory%20Scanners%20Customizing%20Malware%20to%20Evade%20YARA%20PE-sieve%20and%20More.pdf
- https://github.com/opencybersecurityalliance/stix-shifter/tree/develop/adapter-guide/connectors

MORPHISEC

michael@morphisec.com