



Vertex

# Synapse Bootcamp

Module 16

Dynamic Malware Analysis in Synapse

---

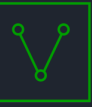
v0.4 - May 2024



# Objectives

- Define dynamic malware analysis
- Identify key data model elements related to dynamic malware analysis
- Understand common pivots and queries to use
- Understand how to use relevant Power-Ups to obtain and enrich data

# What is Dynamic Malware Analysis?



- Examining the "dynamic" properties of a file
- How the file **behaves** when opened / executed
- Does the file:
  - Deliver a payload?
  - Make changes to the host system?
  - Maintain persistence?
  - Collect data?
  - Initiate network communications?
  - Try to evade analysis?
- Often involves a virtual environment or a sandbox analysis service
  - Introduces some challenges
    - Distinguish malware activity from sandbox artifacts



# Threat Intelligence and Malware

- Is this file malicious?
- What does it do / what are its capabilities?
- Have we seen it before?
  - "Similar" samples?
  - Known malware family?
- Do we know who uses it?
  - Type of activity?
  - Known threat group / groups?
- How can we detect it?
  - Unique characteristics / properties?
  - Signature-based detection?
- Can we find related samples / indicators?



# Basic Execution Data

Data	Form
Host (or sandbox) where file executed	<code>it:host</code>
A file "dropped" another file (generic)	<code>file:subfile</code>
An archive file containing other files and associated metadata	<code>file:archive:entry</code>

The amount and type of execution data can vary depending on the source of the data.



# Host / File System Data

Data	Forms
File system activity	<code>it:exec:file:*</code>
Windows registry activity	<code>it:exec:reg:*</code>
Process activity	<code>it:exec:proc</code> <code>it:exec:loadlib, it:exec:mmap, it:exec:thread</code>
Memory activity	<code>it:exec:mutex, it:exec:pipe, etc.</code>

File execution forms are generally **guid** forms. We can record as much (or as little) data as we have.



# Network Data

Data	Forms
Network communications	<code>inet:dns:request</code> <code>it:exec:url</code> <code>inet:http:request</code>
Network connections	<code>inet:flow</code> <code>inet:download</code> <code>inet:urlfile</code> (file hosted at URL)
Opening a port / server	<code>it:exec:bind</code>


Network activity forms are also **guid** forms. We can record as much (or as little) data as we have.



# File Behavior and Guid Forms

- Many details related to writing a file to a host
- May have more (or less) data
- Record what we have (or care about)

Ideally, you will never capture this data "by hand"! Use what is provided by your data sources.

 `it:exec:file:write` [Lift in Research Tool](#) [docs link](#)

An instance of a host writing a file to a filesystem. type: it:exec:file:write  
base: guid

Properties

name	ro	type	doc
:exe		file:bytes	The specific file containing code that wrote to the file. May or may not be the same :exe specified in :proc, if present.
:file		file:bytes	The file that was modified.
:host		it:host	The host running the process that wrote to the file. Typically the same host referenced in :proc, if present.
:path		file:path	The path where the file was written to/modified.
:path:base	—	file:base	The final component of the file path (parsed from :path).
:path:dir	—	file:path	The parent directory of the file path (parsed from :path).
:path:ext	—	str	The file extension of the file name (parsed from :path).
:proc		it:exec:proc	The main process executing code that wrote to / modified the existing file.
:sandbox:file		file:bytes	The initial sample given to a sandbox environment to analyze.
:time		time	The time the file was written to/modified.
.created	—	time	The time the node was created in the cortex.
.seen		ival	The time interval for first/last observation of the node.





# Network Behavior and Guid Forms

- Many details related to network communications
- May have more (or less) data
- Record what we have (or care about)

There is so much detail we could record about an `inet:flow` that it won't fit on one screen capture...

`inet:flow` [Lift in Research Tool](#) [docs link](#)

An individual network connection between a given source and destination. type: `inet:flow`  
base: `guid`

Properties

name	ro	type	doc
<code>:dst</code>		<code>inet:server</code>	The destination address / port for a connection.
<code>:dst:cpes</code>		<code>(it:sec:cpe,)</code>	An array of NIST CPES identified on the destination host.
<code>:dst:exe</code>		<code>file:bytes</code>	The file (executable) that received the connection.
<code>:dst:handshake</code>		<code>str</code>	A text representation of the initial handshake sent by the server.
<code>:dst:host</code>		<code>it:host</code>	The guid of the destination host.
<code>:dst:ipv4</code>		<code>inet:ipv4</code>	The destination IPv4 address.
<code>:dst:ipv6</code>		<code>inet:ipv6</code>	The destination IPv6 address.
<code>:dst:port</code>		<code>inet:port</code>	The destination port.
<code>:dst:proc</code>		<code>it:exec:proc</code>	The guid of the destination process.
<code>:dst:proto</code>		<code>str</code>	The destination protocol.
<code>:dst:softnames</code>		<code>(it:prod:software,)</code>	An array of software names identified on the destination host.
<code>:dst:ssh:key</code>		<code>crypto:key</code>	The key sent by the server as part of an SSH session setup.
<code>:dst:ssl:cert</code>		<code>crypto:x509:cert</code>	The x509 certificate sent by the server as part of an SSL/TLS negotiation.
<code>:dst:txbytes</code>		<code>int</code>	The number of bytes sent by the destination host.
<code>:dst:txcount</code>		<code>int</code>	The number of packets sent by the destination host.
<code>:duration</code>		<code>int</code>	The duration of the flow in seconds.
<code>:from</code>		<code>guid</code>	The ingest source file/iden. Used for reparsing.
<code>:ip:proto</code>		<code>int</code>	The IP protocol number of the flow.
<code>:ip:tcp:flags</code>		<code>int</code>	An aggregation of observed TCP flags commonly provided by flow APIs.



# Dynamic Analysis - Key Properties

- Some secondary properties "link" related execution data
  - o May vary based on data source / Power-Up

Property	Usage
:host	Guid of the host ( <code>it:host</code> ) where execution occurred (may be virtual)
:sandbox:file	The file ( <code>file:bytes</code> ) <b>submitted to the sandbox</b> for analysis May <b>not</b> be the file that performed the action
:exe	The file ( <code>file:bytes</code> ) containing code that <b>performed the action</b> (if known)
:proc	Guid of the process ( <code>it:exec:proc</code> ) that performed the action (if known)



# Detection Data

Detection	Form	Related Forms
Snort signature	<code>it:app:snort:hit</code>	<code>it:app:snort:rule</code>
Antivirus / Antimalware	<code>it:av:scan:result</code> / <code>it:av:prochit (old)</code>	<code>it:av:signature</code> / <code>it:av:sig (old)</code>
YARA rule	<code>it:app:yara:procmatch</code>	<code>it:app:yara:rule</code>
Generic	<code>&lt;(matches) - light edge</code>	<code>meta:rule</code>



# Common Dynamic Analysis Tasks

Question	Workflow
Does this file change anything on the <b>host</b> (e.g., drop files)?	Pivot to host-based execution nodes ( <code>it:exec:file:*</code> ) Pivot to generic <code>file:subfile</code> nodes
Does this file generate <b>network traffic</b> (e.g., communicate with C2)?	Pivot to network-based execution nodes E.g., <code>inet:dns:request</code> , <code>inet:flow</code>
Is this file malicious?	Check for tags on execution artifacts E.g., FQDNs queried, mutexes created
Can I identify other similar files?	Pivot from C2 to other files that use the same C2 Pivot from execution-related properties to find similar files Pivot from detection data to other detected files



# Common Tag Examples

Assessment	Tag Format (Your Assessment)	Example	Third-Party
Is malicious	<code>#cno.mal</code>	<code>#cno.mal</code>	<code>#rep.eset.mal</code>
Associated with a malware family	<code>#cno.mal.&lt;family&gt;</code>	<code>#cno.mal.redtree</code>	<code>#rep.eset.industroyer</code>
Associated with a threat group	<code>#cno.threat.&lt;group&gt;.own</code> <code>#cno.threat.&lt;group&gt;.use</code>	<code>#cno.threat.t872</code> <code>#cno.threat.t872.own</code> <code>#cno.threat.t872.use</code>	<code>#rep.microsoft.nickel</code>
Has certain capabilities or demonstrates use of certain TTPs	<code>#cno.ttp.&lt;category&gt;.&lt;sub&gt;</code> <code>#mitre.attack.&lt;technique&gt;</code>	<code>#cno.ttp.crypt.rc4</code> <code>#cno.ttp.t1573.001</code>	

You can use **triggers** in Synapse to automatically apply tags when certain conditions are met!



# Dynamic Malware Analysis - Demo



# Summary

- **Dynamic malware analysis** involves looking at host activity when a file is opened or executed
  - o File system changes
  - o Registry changes
  - o Network activity
  - o Network-based detection signatures
- Various third-party Power-Ups may provide:
  - o Sandbox execution data
  - o Hashes (or copies) of dropped files
  - o Third-party tags