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	`it:host`
A file "dropped" another file (generic)	`file:subfile`

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	`it:exec:file:*`
Windows registry activity	`it:exec:reg:*`
Process activity	`it:exec:proc` `it:exec:loadlib`, `it:exec:mmap`, `it:exec:thread`
Memory activity	`it:exec:mutex`, `it:exec:pipe`, etc.

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	`inet:dns:request`
	`it:exec:url`
	`inet:http:request`

Network connections	`inet:flow`
	`inet:download`
	`inet:urlfile` (file hosted at URL)

Opening a port / server	`it:exec:bind`

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

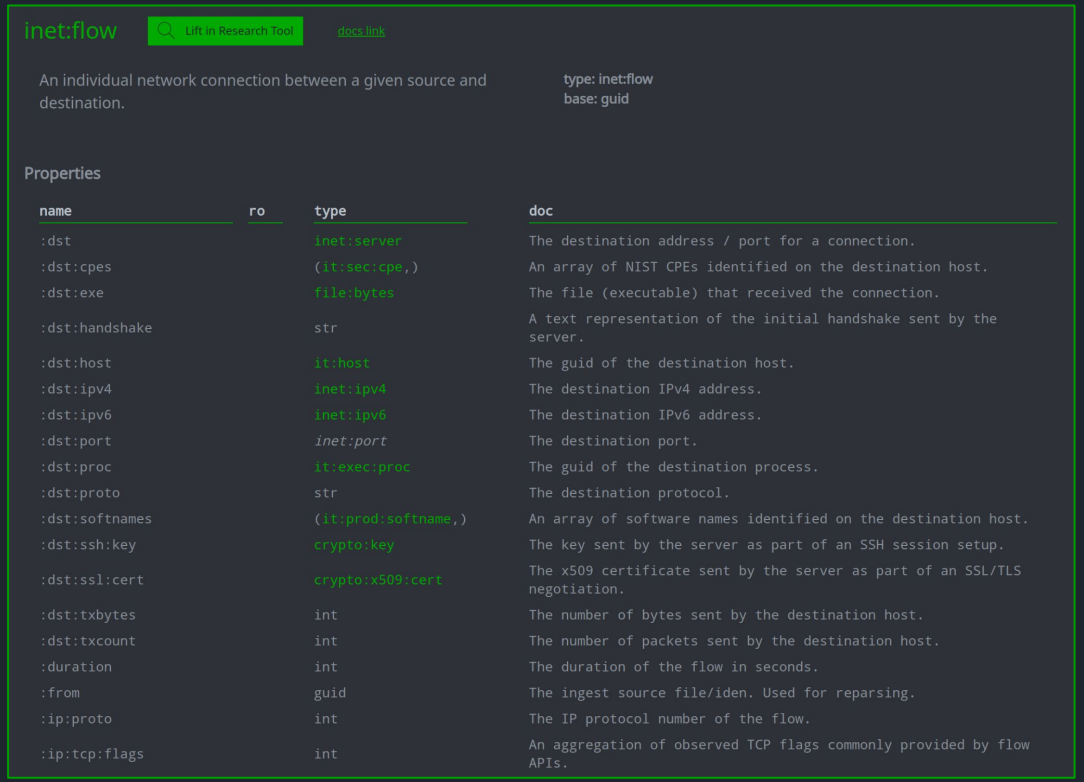
Properties

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

Dynamic Analysis - Key Properties

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

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

Detection Data

Detection	Form	Related Forms
Snort signature	`it:app:snort:hit`	`it:app:snort:rule`
Antivirus / Antimalware	`it:av:scan:result`	`it:av:signame`
YARA rule	`it:app:yara:procmatch`	`it:app:yara:rule`
Generic	`-(matches)>` light edge	`meta:rule`

Common Dynamic Analysis Tasks

Question	Workflow
Does this file change anything on the **host** (e.g., drop files)?	Pivot to host-based execution nodes (`it:exec:file:*`) Pivot to generic `file:subfile` nodes
Does this file generate **network traffic** (e.g., communicate with C2)?	Pivot to network-based execution nodes E.g., `inet:dns:request`, `inet:flow`
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	`#cno.mal`	`#cno.mal`	`#rep.eset.mal`
Associated with a malware family	`#cno.mal.<family>`	`#cno.mal.redtree`	`#rep.eset.industroyer`

Associated with a threat group	`#cno.threat.<group>.own`
	`#cno.threat.<group>.use`
		`#cno.threat.t872`
		`#cno.threat.t872.own`
		`#cno.threat.t872.use`
			`#rep.microsoft.nickel`

Has certain capabilities or demonstrates use of certain TTPs	`#cno.ttp.<category>.<sub>`	`#cno.ttp.crypt.rc4`	

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
  - File system changes
  - Registry changes
  - Network activity
  - Network-based detection signatures
- Various third-party Power-Ups may provide:
  - Sandbox execution data
  - Hashes (or copies) of dropped files
  - Third-party tags