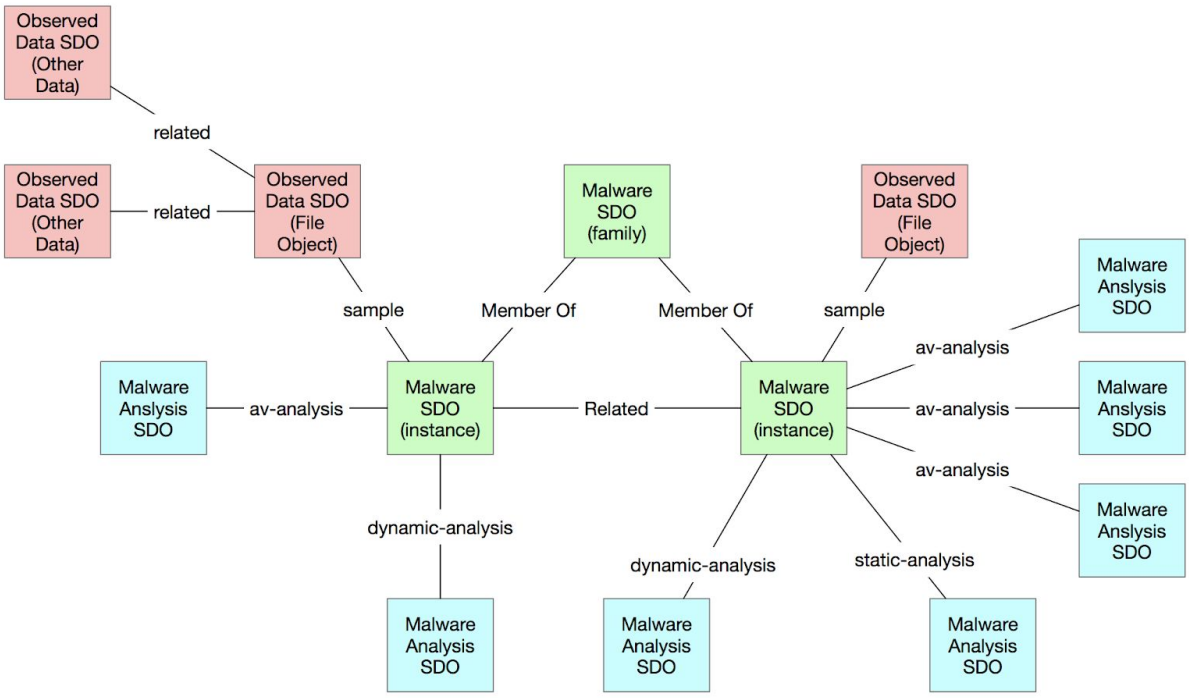

2.8 Malware

Type Name: `malware`

Malware characterizes Malware Instances or Malware Families, including identifying information, metadata, and data that may be derived from various forms of analysis.

It is important to note that this is not a substitute for an Indicator SDO and is not intended to provide detection capabilities for malware instances or families. While there may be some overlap in terms of the data contained in this SDO and the Indicator SDO, we strongly encourage the use of STIX Indicators for the detection of malware entities, due to its use of the STIX Patterning language and the clear semantics that it provides.

2.8.1 Properties

Common Properties
`type`, **`spec_version`**, **`id`**, **`created_by_ref`**, **`created`**, **`modified`**, **`revoked`**, **`labels`**, **`confidence`**, **`lang`**, **`external_references`**, **`object_marking_refs`**, **`granular_markings`**
Malware Specific Properties

is_family, **name**, **description**, **malware_types**, **aliases**, **kill_chain_phases**, **first_seen**, **last_seen**, **os_execution_envs**, **architecture_execution_envs**, **implementation_languages**, **capabilities**, **sample_ref**		
Property Name	**Type**	**Description**
type (required)	`string`	The value of this field **MUST** be `malware`
is_family (required)	`boolean`	Whether the object represents a malware family (if `true`) or a malware instance (if `false`).
name (required)	`string`	A name used to identify the malware instance or family, as specified by the producer of the SDO. If a name for a malware instance is not available, the SHA-256 hash value or binary filename **SHOULD** be used instead.
description (optional)	`string`	A description that provides more details and context about the malware instance or family, potentially including its purpose and its key characteristics.
malware_types (required)	`list` of type `open-vocab`	This property is an open vocabulary that specifies a set of categorizations for the malware being described. This is an open vocabulary and values **SHOULD** come from the `malware-type-ov` vocabulary.
aliases (optional)	`list` of type `string`	Alternative names used to identify this Malware.
kill_chain_phases (optional)	`list` of type `kill-chain-phase`	The list of Kill Chain Phases for which this malware instance or family can be used.
first_seen (optional)	`timestamp`	The time that the malware family or malware instance was first seen. This property is a summary property of data from sightings and other data that may or may not be available in STIX. If new sightings are received that are earlier than the first seen timestamp, the object may be updated to account for the new data.

last_seen (optional)	`timestamp`	The time that the malware family or malware instance was last seen. This **MUST** be greater than or equal to the timestamp in the **first_seen** property. This property is a summary property of data from sightings and other data that may or may not be available in STIX. If new sightings are received that are earlier than the first seen timestamp, the object may be updated to account for the new data.
os_execution_envs (optional)	`list` of type `string`	The operating systems that the malware family or malware instance is executable on. Each string value for this property **MUST** be a CPE v2.3 entry from the official NVD CPE Dictionary [NVD] .
architecture_execution_envs (optional)	`list` of type `open-vocab`	The processor architectures (e.g., x86, ARM, etc.) that the malware family or malware instance is executable on. This is an open vocabulary and values **SHOULD** come from the `processor-architecture-ov` vocabulary.
implementation_languages (optional)	`list` of type `open-vocab`	The programming language(s) used to implement the malware family or malware instance. This is an open vocabulary and values **SHOULD** come from the `implementation-language-ov` vocabulary.
capabilities (optional)	`list` of type `open-vocab`	Specifies any capabilities identified for the malware instance (or family). This is an open vocabulary and values **SHOULD** come from the `malware-capabilities-ov` vocabulary.
sample_ref (optional)	`identifier`	The **sample_ref** property specifies the ID of the Observed Data object that contain the Cyber Observable `file` object for this instance of malware.

		Other Observed Data objects can be related to this ID. Those Observed Data objects could describe any basic identifying data (e.g., filename, hashes, etc.) extracted from the binaries associated with the malware instance (or family). The ID can also point to the Observed Data objects that contain the actual binaries that are included as a base64-encoded payload, via a reference to an Artifact Object using the **`content_ref`** property. To minimize the risk of a consumer compromising their system in parsing malware samples, producers are **strongly encouraged** to share zipped, password-protect samples instead of raw, base64-encoded samples.

2.8.2 Relationships

These are the relationships explicitly defined between the Malware object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from this object by way of the Relationship Object. The reverse relationships (relationships "to" this object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship name or, as with open vocabularies, user-defined names.

Embedded Relationships			
`created_by_ref`	`identifier` (of type `identity`)		
`object_markings_refs`	`identifier` (of type `marking-definition`)		
Common Relationships			
`duplicate-of`, `derived-from`, `related-to`			
Source	**Name**	**Target**	**Description**
`malware`	`authored-by`	`threat-actor`	This Relationship describes that the malware instance (or family) was developed by the related threat actor.

malware	delivered-by	attack-pattern, infrastructure, tool, observed-data	This Relationship describes that the malware instance (or family) was delivered by malicious infrastructure tool, or as part of a particular attack pattern, or some non-malware entity such as an email as characterized by observed-data.
malware	targets	identity, location	This Relationship documents that a malware instance (or family) is being used to target an Identity or Location or exploit a Vulnerability. For malware families, this can be used to capture the full set of identities, locations, or vulnerabilities targeted by the family. Similarly, a targets Relationship linking a malware instance (or family) representing a downloader to an Identity representing the energy sector means that downloader is typically used against targets in the energy sector.
malware	uses	attack-pattern	This Relationship documents that this malware instance (or family) uses the attack pattern to achieve its objectives.
malware	exploits	vulnerability	This Relationship documents that this malware instance (or family) exploits or attempts to exploit a particular vulnerability. For example, a exploits Relationship linking a malware instance (or family) representing a downloader to a Vulnerability for CVE-2016-0001 means that the malware instance (or family) exploits that vulnerability.

malware	uses	tool, malware	This Relationship documents that this malware instance (or family) uses the tool to perform its functions.
malware	controls	malware	This Relationship documents that this malware instance (or family) can control other malware which may be resident on the same system on which it is executing. Note that this is not meant to imply or state that the malware instance (or family) drops other malware (which is covered by the dropped-by relationship). Rather, it is meant to state that the malware instance (or family) is able to subvert or control other malware to achieve its goals.
malware	variant-of	malware	This Relationship is used to document that one malware instance (or family) is a variant of another malware instance (or family). Only the following uses of this relationship are valid: malware instance → malware family: a malware instance is a variant of a malware family. For example, a particular Zeus version 2 sample is a variant of the broader Zeus family. malware family → malware family: a malware family is a variant of another malware family. For example, the Gameover Zeus family is a variant of the broader Zeus family. malware instance → malware instance: a malware instance is a variant of another malware instance.

			For example, a particular Cryptolocker instance that is based on an another Cryptolocker instance with minor changes.
malware	drops	malware, tool	This Relationship covers the case where a malware instance drops or downloads another malware instance or a tool. This is especially common with “first-stage” malware instances such as downloaders.
Reverse Relationships			
indicator	indicates	malware	See forward relationship for definition.
course-of-action	mitigates	malware	See forward relationship for definition.
campaign, intrusion-set, threat-actor, attack-pattern	uses	malware	See forward relationship for definition.
tool	drops	malware	See forward relationship for definition.
malware-analysis	av-analysis, static-analysis, dynamic-analysis	malware	See forward relationship for definition.

2.9 Malware Analysis

Type Name: `malware-analysis`

Malware Analysis captures the metadata and results of a particular analysis performed (static or dynamic) on the malware instance or family. At least one of **`start_time`**, **`end_time`**, **`analysis_tools`**, **`analysis_environment`**, or **`results`** **MUST** be included when using this type.

2.9.1 Properties

Common Properties

type, **spec_version**, **id**, **created_by_ref**, **created**, **modified**, **version**, **revoked**,, **external_references**, **object_markings_refs**, **granular_markings**		
Malware Analysis Specific Properties		
product, **engine_version**, **definition_version**, **submitted**, **scanned**, **av_result**, **details**, **analysis_tools**, **analysis_environment**, **results**		
Property Name	**Type**	**Description**
product (required)	`string`	The name of the AV engine or product that was used. Product names **SHOULD** be all lower-case with words separated by a dash "-". For cases where the name of a product cannot be specified, a value of "anonymized" field **MUST** be used.
engine_version (optional)	`string`	The version of the AV engine or product that was used to perform the AV scan or analysis.
definition_version (optional)	`string`	The version of the AV definitions used by the AV scanner tool.
submitted (optional)	`timestamp`	The date and time that the malware was first submitted for scanning or analysis. This value will stay constant while the scanned date can change.
scanned (optional)	`timestamp`	The date and time that the malware was scanned or analysed. This field can be used to capture how a scan changes over time.
av_result (required)	`string`	The classification result or name assigned to the malware instance by the AV scanner tool. If no resulting context-specific classification value or name is provided by the AV scanner tool or cannot be specified then the result value **SHOULD** come from the `malware-av-result-ov` open vocabulary.
analysis_environment (optional)	`dictionary`	A description of the analysis environment used for the analysis, if applicable. Each key name **SHOULD** come from the `malware-analysis-environment-ov` open vocabulary. Each corresponding key

		value **MUST** meet the constraints specified for the key in the vocabulary. For cases where the key name does not come from an included vocabulary, the value **MUST** be a valid STIX common data type.
details (required)	`dictionary`	The results of the analysis, as a set of key value pairs. Each key name **SHOULD** come from the `malware-dynamic-analysis-data-ov` OR `malware-static-analysis-data-ov` open vocabularies. Each corresponding key value **MUST** meet the constraints specified for the key in the vocabulary. For cases where the key name does not come from an included vocabulary, the value **MUST** be a valid STIX common data type.

2.9.2 Relationships

These are the relationships explicitly defined between the Malware Analysis object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from this object by way of the Relationship Object. The reverse relationships (relationships "to" this object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship name or, as with open vocabularies, user-defined names.

Embedded Relationships	
created_by_ref	`identifier` (of type `identity`)
object_markings_refs	`identifier` (of type `marking-definition`)

Common Relationships			
`duplicate-of`, `derived-from`, `related-to`			
Source	**Name**	**Target**	**Description**
`malware-analysis`	`av-analysis`	`malware`	This Relationship describes that the malware analysis is AV scan results for related malware.
`malware-analysis`	`static-analysis`	`malware`	This Relationship describes that the malware analysis is

			static analysis results for related malware.
malware-analysis	dynamic-analysis	malware	This Relationship describes that the malware analysis is dynamic analysis results for related malware.
Reverse Relationships			

7 Vocabularies

7.8 Malware Analysis Environment

Vocabulary Name: malware-analysis-environment-ov

The malware analysis environment vocabulary is currently used in the following SDO(s):

- Malware

This is an open vocabulary that covers various aspects of the environment used in the dynamic analysis of a malware instance or family, such as the type of virtual machine used to host the guest operating system in which the malware was executed.

Vocabulary Summary	
host-vm, installed-software, operating-system	
Vocabulary Value	**Description**
host-vm	The virtual machine used to host the guest operating system (if applicable) used for the the dynamic analysis of the malware instance or family. If this value is not included in conjunction with operating-system, this means that the dynamic analysis was performed on bare metal (i.e., without virtualization). The corresponding value property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type software.
installed-software	Any non-standard software installed on the operating system (specified through the operating-system value) used for the dynamic analysis of the malware instance or family.

	The corresponding value property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `software`.
`operating-system`	The operating system used for the dynamic analysis of the malware instance or family. This applies to virtualized operating systems as well as those running on bare metal. The corresponding value property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `software`.

7.9 Malware Capabilities

Vocabulary Name: `malware-capabilities-ov`

The malware capabilities vocabulary is currently used in the following SDO(s):

- Malware

This is an open vocabulary that covers common capabilities that may be exhibited by a malware instance or family.

Vocabulary Summary	
`accesses-remote-machines`, `anti-debugging`, `anti-disassembly`, `anti-emulation`, `anti-memory-forensics`, `anti-sandbox`, `anti-vm`, `captures-input-peripherals`, `captures-output-peripherals`, `captures-system-state-data`, `cleans-traces-of-infection`, `commits-fraud`, `communicates-with-c2`, `compromises-data-availability`, `compromises-data-integrity`, `compromises-system-availability`, `controls-local-machine`, `degrades-security-software`, `degrades-system-updates`, `determines-c2-server`, `emails-spam`, `escalates-privileges`, `evades-av`, `exfiltrates-data`, `fingerprints-host`, `hides-artifacts`, `hides-executing-code`, `infects-files`, `infects-remote-machines`, `installs-other-components`, `persists-after-system-reboot`, `prevents-artifact-access`, `prevents-artifact-deletion`, `probes-network-environment`, `self-modifies`, `steals-authentication-credentials`, `violates-system-operational-integrity`	
Vocabulary Value	**Description**
`accesses-remote-machines`	Indicates that the malware instance or family is able to access one or more remote machines.
`anti-debugging`	Indicates that the malware instance or family is able to prevent itself from being debugged and/or from being run in a debugger or is able to make debugging more difficult.

`anti-disassembly`	Indicates that the malware instance or family is able to prevent itself from being disassembled or make disassembly more difficult.
`anti-emulation`	Indicates that the malware instance or family is able to prevent its execution inside of an emulator or is able to make emulation more difficult.
`anti-memory-forensics`	Indicates that the malware instance or family is able to prevent or make memory forensics more difficult.
`anti-sandbox`	Indicates that the malware instance or family is able to prevent sandbox-based behavioral analysis or make it more difficult.
`anti-vm`	Indicates that the malware instance or family is able to prevent virtual machine (VM) based behavioral analysis or make it more difficult.
`captures-input-peripherals`	Indicates that the malware instance or family is able to capture data from a system's input peripheral devices, such as a keyboard or mouse. This includes things like keylogging.
`captures-output-peripherals`	Indicates that the malware instance or family captures data sent to a system's output peripherals, such as a display. Examples include things like screen scraping.
`captures-system-state-data`	Indicates that the malware instance or family is able to capture information about a system's state (e.g., data currently in its RAM).
`cleans-traces-of-infection`	Indicates that the malware instance or family is able to clean traces of its infection (e.g., file system artifacts) from a system.
`commits-fraud`	Indicates that the malware instance or family commits fraud, such as click fraud (for example).
`communicates-with-c2`	Indicates that the malware instance or family is able to communicate (i.e., send or receive data) with a command and control (C2) server.

compromises-data-availability	Indicates that the malware instance or family is able to compromise the availability of data on the local system on which it is executing and/or one or more remote systems. For example, encrypting data on disk, as done by ransomware.
compromises-data-integrity	Indicates that the malware instance or family is able to compromise the integrity of some data that resides on (e.g., in the case of files) or is received/transmitted (e.g., in the case of network traffic) by the system on which it is executing.
compromises-system-availability	Indicates that the malware instance or family is able to consume system resources for its malicious purposes, such as password cracking or participating in a DDoS botnet, thereby compromising the availability of the local system and/or one or more remote systems.
controls-local-machine	Indicates that the malware instance or family is able to control the machine on which it is executing (e.g., RATs).
degrades-security-software	Indicates that the malware instance or family is able to bypass or disable security programs or operating system security features on a system (including mobile devices), either by stopping them from executing or by making changes to their code or configuration parameters. For example, malware that blocks the local machine from accessing the websites of security vendors.
degrades-system-updates	Indicates that the malware instance or family is able to disable the downloading and installation of system updates and patches.
determines-c2-server	Indicates that the malware instance or family is able to identify one or more command and control (C2) servers with which to communicate (e.g., DGA).
emails-spam	Indicates that the malware instance or family is able to send spam email messages.
escalates-privileges	Indicates that the malware instance or family is able to escalate the privileges under which it is executing.

evades-av	Indicates that the malware instance or family is able to evade detection by antivirus tools.
exfiltrates-data	Indicates that the malware instance or family is able to gather, prepare, (possibly obfuscate) data and transmit it to exfiltration points.
fingerprints-host	Indicates that the malware instance or family is able to fingerprint or probe the configuration of the host system on which it is executing for the purpose of altering its behavior based on this environment.
hides-artifacts	Indicates that the malware instance or family is able to hide its artifacts, such as files and open ports.
hides-executing-code	Indicates that the malware instance or family is able to hide its code by compromising the bootloader, kernel modules, hypervisor, etc.
infects-files	Indicates that the malware instance or family is able to infect one or more files on the system on which it executes. For example, malware which injects a malicious payload into all PDFs on a host as a means of propagation.
infects-remote-machines	Indicates that the malware instance or family is able to self-propagate to a remote machine or infect a machine with malware that is different than itself.
installs-other-components	Indicates that the malware instance or family is able to install additional components. This encompasses the dropping/downloading of other malicious components such as libraries, other malware, and tools.
persists-after-system-reboot	Indicates that the malware instance or family is able to continue executing after the reboot of the system on which it is resident.
prevents-artifact-access	Indicates that the malware instance or family is able to prevent its artifacts (e.g., files, registry keys, etc.) from being accessed.
prevents-artifact-deletion	Indicates that the malware instance or family is able to prevent its artifacts (e.g., files, registry keys, etc.) from being deleted.

probes-network-environment	Indicates that the malware instance or family is able to probe the properties of its network environment, e.g. to determine whether it funnels traffic through a proxy.
self-modifies	Indicates that the malware instance or family is able to modify itself.
steals-authentication-credentials	Indicates that the malware instance is able to steal authentication credentials.
violates-system-operational-integrity	Indicates that the malware instance or family is able to compromise the operational integrity of the system on which it is executing and/or one or more remote systems, e.g., by causing them to operate beyond their set of specified operational parameters. For example, malware that causes the CPU fan on the machine that it is executing to spin at a higher than normal speed.

7.10 Malware Dynamic Analysis Data

Vocabulary Name: malware-dynamic-analysis-data-ov

The dynamic malware analysis data vocabulary is currently used in the following SDO(s):

- Malware

This is an open vocabulary that covers various common types of analysis data that may be obtained through the dynamic analysis of a malware instance or family.

Vocabulary Summary	
created-files,	
Vocabulary Value	**Description**
created-files	Any files that were created by the malware instance or known to be created by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type file.
created-mutexes	Any mutexes that were created by the malware instance or known to be created by the malware family.

	The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `mutex`.
`created-processes`	Any processes that were created by the malware instance or known to be created by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `process`.
`loaded-dlls`	Any DLLs that were loaded by the malware instance or known to be loaded by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `file`.
`loaded-services`	Any Windows services that were loaded by the malware instance or known to be loaded by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `process` and **MAY** have an extension of `windows-service-ext`.
`read-processes`	Any processes that were read from by the malware instance or known to be read from by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `process`.
`terminated-processes`	Any processes that were terminated by the malware instance or known to be terminated by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `process`.
`written-processes`	Any processes that were written to by the malware instance or known to be written to by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `process`.
`opened-files`	Any files that were opened by the malware instance or known to be opened by the malware family.

	The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `file`.
`deleted-files`	Any files that were deleted by the malware instance or known to be deleted by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `file`.
`read-files`	Any files that were read from by the malware instance or known to be read from by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `file`.
`written-files`	Any files processes that were written to by the malware instance or known to be written to by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `fileprocess`.
`created-directories`	Any directories that were created by the malware instance or known to be created by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `directory`.
`written-directories`	Any directories that were written to by the malware instance or known to be written to by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `directory`.
`created-registry-keys`	Any Windows registry keys that were created by the malware instance or known to be created by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `windows-registry-key`.
`deleted-registry-keys`	Any Windows registry keys that were deleted by the malware instance or known to be deleted by the malware family.

	The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `windows-registry-key`.
`opened-registry-keys`	Any Windows registry keys that were opened by the malware instance or known to be opened by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `windows-registry-key`.
`written-registry-key-values`	Any Windows registry key values that were written or modified by the malware instance or known to be written or modified by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `windows-registry-key`.
`read-registry-keys`	Any Windows registry key values that were read by the malware instance or known to be read by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `windows-registry-key`.
`dns-queries`	Any DNS queries that were performed by the malware instance or known to be performed by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `network-traffic` and **MUST** have an extension of `dns-query-ext`.
`http-requests`	Any HTTP requests that were performed by the malware instance or known to be performed by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `network-traffic` and **MUST** have an extension of `http-request-ext`
`contacted-domains`	Any domains that contacted by the malware instance or known to be contacted by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `domain-name`.
`contacted-urls`	Any URLs that were contacted by the malware instance or known

	to be contacted by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `url`.
`contacted-ips`	Any IP addresses that contacted by the malware instance or known to be contacted by the malware family. The corresponding value in the **dynamic_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `ipv4-addr` or of type `ipv6-addr`.

7.11 Malware Static Analysis Data

Vocabulary Name: `malware-static-analysis-data-ov`

The static malware analysis data vocabulary is currently used in the following SDO(s):

- Malware

This is an open vocabulary that covers various common types of analysis data that may be obtained through the static analysis of a malware instance or family.

Vocabulary Summary	
`addresses`, `c2-addresses`, `c2-socket-addresses`, `c2-urls`, `certificates`, `credentials`, `directories`, `file-headers`, `file-names`, `file-paths`, `injection-processes`, `intervals`, `keys`, `listen-ports`, `mission-ids`, `mutexes`, `output-files`, `packers`, `ports`, `registry-keys`, `services`, `socket-addresses`, `strings`, `urls`, `user-agents`, `usernames`, `versions`	
Vocabulary Value	**Description**
`addresses`	An IP address or domain name used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `ipv4-addr`, `ipv6-addr`, or `domain-name`.
`c2-addresses`	An IP address or domain name associated with command and control of the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file.

	The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `ipv4-addr`, `ipv6-addr`, or `domain-name`.
`c2-socket-addresses`	A socket address used by the malware instance or family for command and control. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `socket-addr`.
`c2-urls`	A URL associated with command and control of the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `url`.
`certificates`	Any certificates used to sign the binaries associated with the malware instance or family or to encrypt its network traffic. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `x509-certificate`.
`credentials`	A credential used by the malware instance or family, as a tuple of username and password. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `user-account`.
`directories`	A directory name used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `directory`.
`file-headers`	Specifies any file header data (e.g., PE binary headers) extracted from the binaries associated with the malware instance or family. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `file`.

file-names	A filename used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type file.
file-paths	A file system path used by the malware instance or family, including both a directory and a filename. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type file.
injection-processes	A process into which the malware instance or family is injected. Usually this is a process name but it may take other forms such as a filename of the executable. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type file OR process.
intervals	A duration of time that the malware instance or family waits between beacons or other activity, in seconds. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type integer.
keys	An encryption, encoding, or obfuscation key used by the malware instance or family. When this represents binary data, it should be hex encoded with no other markup. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type string.
listen-ports	A TCP or UDP port that is opened for listening by the malware instance or family. This has the same format as port but indicates an incoming connection where the malware is the server. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type socket-addr.

mission-ids	An attacker specified identifier encoded in the malware instance or family, usually reflected in beacons and often related to target or time of attack. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type string.
mutexes	A mutex name used to prevent multiple executions of the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type mutex.
output-files	A relevant or related file created during parsing of the malware instance or family, including a filename and hash. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type file.
packers	Any executable packers used to pack the binaries associated with the malware instance or family. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type software.
ports	A combination of a port number and protocol, as used by the malware instance or family. This generally refers to outbound connections where the malware is the client. Other network layer protocols, such as ICMP can be represented here. Application layer protocols, such as HTTP, should be indicated via other entries such as url. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type observable-objects. Each base object in the container **MUST** be of type socket-addr.
registry-keys	A registry key created or used by the malware instance or family, which may include a value name and data. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file.

	The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `windows-registry-key`.
`services`	A Windows service created or used by the malware instance or family, including the display name, description, service image, and dll. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `process` and MUST include the `windows-service-ext` extension.
`socket-addresses`	A socket address used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `socket-addr`.
`strings`	Specifies any strings extracted from the binaries associated with the malware instance or family. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `list`, and each entry in the list **MUST** be of type `string`.
`urls`	A URL used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `url`.
`user-agents`	A software identifier used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `list`. Each list entry, corresponding to a single user agent, **MUST** be of type `string`.
`usernames`	A username used by the malware instance or family. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type `observable-objects`. Each base object in the container **MUST** be of type `user-account`.

versions	The version of the malware instance or family, to include build version. To the degree possible this should be based directly on artifacts from the malware. This data is commonly stored as a configuration parameter embedded in the malware binary or in a separate configuration file. The corresponding value in the **static_analysis_data** property for this item **MUST** be of type string.

7.12 Malware Implementation Language

Vocabulary Name: malware-implementation-language-ov

The implementation language vocabulary is currently used in the following SDO(s):

- Malware

This is a non-exhaustive, open vocabulary that covers common programming languages and is intended to characterize the languages that may have been used to implement a malware instance or family.

Vocabulary Summary	
applescript, bash, c++, c#, go, java, javascript, lua, objective-c, php, powershell, python, scala, swift, typescript, x86-32, x86-64	
Vocabulary Value	**Description**
applescript	Specifies the AppleScript programming language.
bash	Specifies the Bash programming language.
c++	Specifies the C++ programming language.
c#	Specifies the C# programming language.
go	Specifies the Go (sometimes referred to as golang) programming language.
java	Specifies the JAVA programming language.
javascript	Specifies the JavaScript programming language.
lua	Specifies the Lua programming language.
objective-c	Specifies the Objective-C programming language.
php	Specifies the PHP programming language.

powershell	Specifies the Windows Powershell programming language.
python	Specifies the Python programming language.
scala	Specifies the Scala programming language.
swift	Specifies the Swift programming language.
typescript	Specifies the TypeScript programming language.
x86-32	Specifies the x86 32-bit Assembly programming language.
x86-64	Specifies the x86 64-bit Assembly programming language.

7.13 Malware Processor Architecture

Vocabulary Name: malware-processor-architecture-ov

The processor architecture vocabulary is currently used in the following SDO(s):

- Malware

This is a non-exhaustive, open vocabulary that covers common processor architectures and is intended to characterize the architectures that a malware instance or family may be able to execute on.

Vocabulary Summary	
alpha, arm, ia-64, mips, powerpc, sparc, x86, x86-64	
Vocabulary Value	**Description**
alpha	Specifies the Alpha architecture.
arm	Specifies the ARM architecture.
ia-64	Specifies the 64-bit IA (Itanium) architecture.
mips	Specifies the MIPS architecture.
powerpc	Specifies the PowerPC architecture.
sparc	Specifies the SPARC architecture.
x86	Specifies the 32-bit x86 architecture.
x86-64	Specifies the 64-bit x86 architecture.

Malware AV Result

Vocabulary Name: `malware-av-result-ov`

The processor architecture vocabulary is currently used in the following SDO(s):

- Malware

This is a non-exhaustive, open vocabulary that captures common types of generic malware anti-virus (AV) tool results.

Vocabulary Summary	
`malicious`, `suspicious`, `benign`, `unknown`	
Vocabulary Value	**Description**
`malicious`	The AV tool reported the malware binary as malicious.
`suspicious`	The AV tool reported the malware binary as suspicious but not definitively malicious.
`benign`	The AV tool reported the malware binary as benign.
`unknown`	The AV tool was unable to determine whether the malware binary is malicious.