URLhaus | Data exposed through Real Time Intelligence Feed v. publicly available data

URLhaus provides context-rich signal from abuse.ch, informing of malicious URLs being used for malware distribution.

Users can access this data via the Real Time Intelligence Feed, or via publicly available APIs. These access methods provide different metadata outputs, as highlighted in this document.

The Real Time Intelligence Feed is the only access method to receive all observed threat signals.

Metadata field	Metadata description	Real Time Feed	Publicly available data
URL Additions			
_idx	An integer representing the incremental number of the message.	✓	✗
_ts	The Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓ *
uuid	An internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message.	✓	✓
id	Represents the ID of the URL in the URLhaus database. It uniquely identifies the specific URL tracked. It also can be used to assemble the HTTP link to the URLhaus record page (https:// urlhaus.abuse.ch/url/id/).	✓	✓
url	Is the added URL.	✓	✓
host	The host associated with this URL (extracted from the URL).	✓	✓
url_status	Is a string that represents the status of the URL. Possible values are 'online', 'offline', and 'unknown'. 'unknown' is reported when the URL has not yet been checked by URLhaus.	✓	✓
anonymous	Is a boolean value indicating if the reporter of the URL wants to stay anonymous.	✓	✓
reporter	Is the handle of the reporter of the URL or 'null' if it should be anonymous. Currently, the handle equals the Twitter handle of the reporter. After migration to a new authentication system for abuse.ch, this handle will change to one from abuse.ch's own authentication platform.	✓	✓
tags	Are a list of tags associated with the added URL, as shown in URLhaus. Tags are "free field" and defined by the reporter (submitter) for the URL.	✓	✓
URL Removals			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the realtime infrastructure.	✓	✗
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message.	✓	✗
id	Represents the ID of the URL in the URLhaus database. This is needed to assemble the HTTP link to the URLhaus record page.	✓	✗
url	Is the URL being added.	✓	✗
removal_note	Is a text string, human-readable, that describes why the URL has been removed.	✓	✗
URL Changes			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message.	✓	✗
id	Represents the ID of the URL in the URLhaus database. This is needed to assemble the HTTP link to the URLhaus record page (https:// urlhaus.abuse.ch/url/id/).	✓	✗
url	Is the URL being modified.	✓	✗
field	Shows which field has been changed. Fields currently supported are: tag, url_status	✓	✗
value	Is the new value of the affected field.	✓	✗
action	This represents what action happened to the field. The action could be add, remove or change.	✓	✗
New file download			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Describes the type of this message and is always "file_download".	✓	✗
sha256_hash	Is the SHA256 hash of the file.	✓	✓
md5_hash	Is the MD5 hash of the file.	✓	✓
Observed payloads			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Describes the type of this message and is always "payload_observed".	✓	✗
id	Represents the ID of the URL in the URLhaus database. This is needed to assemble the HTTP link to the URLhaus record page (https:// urlhaus.abuse.ch/url/id/).	✓	✗
url	Is the full URL from which the file was downloaded.	✓	✓
mime_type	Is the Multipurpose Internet Mail Extensions (MIME) type of the payload received.	✓	✗
file_type	Is the result of the Unix "file" command (not to be confused with the content-type header from the webserver).	✓	✓
file_ext	Is the guessed file extension (or 'null', if not available).	✓	✗
file_size	Is the size (in bytes) of the payload received.	✓	✗
file_name	Is the filename as extracted from the HTTP Content-Disposition header in the response. It's 'null' if the info is not available.	✓	✗
md5_hash	Is the MD5 hash of the payload received.	✓	✓
sha256_hash	Is the SHA256 hash of the payload received.	✓	✓
imphash	Is the imphash of the payload received.	✓	✗
ssdeep	Is the ssdeep of the payload received.	✓	✗
tlsh	Is the tlsh of the payload received.	✓	✗
telfhash	Is the telfhash of the payload received.	✓	✗
malware	This is the malware family.	✓	✓
Payload changes			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Describes the type of this message and is always "payload_change".	✓	✗
md5_hash	Is the MD5 hash of the payload received.	✓	✗
sha256_hash	Is the SHA256 hash of the payload received.	✓	✗
field	Shows the affected field where the change occurred. Currently, only malware is supported.	✓	✗
value	Is the new value of the affected field.	✓	✗
action	This represents what action happened to the field. The action could be add, remove or change.	✓	✗

*** Comparable data for this field is provided via publicly available data, however the field names are not an exact match.**
If you require the comparable field name, please speak with your Spamhaus contact.

MalwareBazaar | Data exposed through Real Time Intelligence Feed v. publicly available data

MalwareBazaar provides context-rich signal from abuse.ch, providing intelligence on confirmed malware samples.

Users can access this data via the Real Time Intelligence Feed, or via publicly available APIs. These access methods provide different metadata outputs, as highlighted in this document.

The Real Time Intelligence Feed is the only access method to receive all observed threat signals.

Metadata field	Metadata description	Real Time Feed	Publicly available data
File Additions			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓*
`uuid`	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
`type`	Defines the type of message. It's always 'file_addition'.	✓	✓*
`file_size`	Is the size (in bytes) of the payload received.	✓	✓
`file_name`	Is the filename as extracted from the HTTP `Content-Disposition` header in the response.	✓	✓
`md5_hash`	Is the MD5 hash of the payload received.	✓	✓
`sha256_hash`	Is the SHA256 hash of the payload received.	✓	✓
`sha1_hash`	Is the SHA1 hash of the file.	✓	✓
`sha3_384_hash`	Is the SHA3-384 hash of the file.	✓	✓
`humanhash`	Is the human-readable hash. Provides human-readable representations of digests.	✓	✗
`imphash`	Is the imphash of the payload received.	✓	✓
`ssdeep`	Is the ssdeep of the payload received.	✓	✓
`tlsh`	Is the tlsh of the payload received.	✓	✓
`telfhash`	Is the telfhash of the payload received.	✓	✓
`gimphash`	Is the gimphash of the file.	✓	✓
`dhash_icon`	Is the dhash of the file icon.	✓	✓
`mime_type`	Is the Multipurpose Internet Mail Extensions (MIME) type of the payload received.	✓	✓*
`file_type`	Is the result from Unix "file" command.	✓	✓*
`file_ext`	Is the guessed file extension (or 'null', if not available).	✓	✓*
`malware`	This is the malware family.	✓	✓*
`tags`	Is a list of tags associated with this file.	✓	✓
`anonymous`	Is a boolean that indicates whether the submitter of this file wants to remain anonymous or not.	✓	✓
`reporter`	Is the abuse.ch handle of the submitter of this file (or 'null', if not available).	✓	✓
`origin_country`	Two letter Country code of the country from where the submission has been made.	✓	✓
`delivery_method`	Distributed via e-mail attachment.	✓	✓
`comment`	Is a comment from the reporter of the URL (or 'null', if not available).	✓	✓
File changes			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
`uuid`	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
`type`	Defines the type of message. It's always 'file_change'.	✓	✗
`md5_hash`	Is the MD5 hash of the payload received.	✓	✗
`sha256_hash`	Is the SHA256 hash of the payload received.	✓	✗
`sha1_hash`	Is the SHA1 hash of the file.	✓	✗
`sha3_384_hash`	Is the SHA3-384 hash of the file.	✓	✗
`field`	Shows the affected field where the change occurred (supported fields: `tag`, `malware`, `file_ext`).	✓	✗
`value`	Is the new value of the affected field.	✓	✗
`action`	Is an enumerated field that describes the action. May contain `add`, `remove`, `change`.	✓	✗
File removals			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
`uuid`	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
`type`	Defines the type of message. It's always 'file_removal'.	✓	✗
`md5_hash`	Is the MD5 hash of the payload received.	✓	✗
`sha256_hash`	Is the SHA256 hash of the payload received.	✓	✗
`sha1_hash`	Is the SHA1 hash of the file.	✓	✗
`sha3_384_hash`	Is the SHA3-384 hash of the file.	✓	✗
`removal_note`	Is a text string showing the removal note as inserted by the system or the remover.	✓	✗
YARA matches			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓*
`uuid`	UUID identifying this message.	✓	✗
`type`	Type of this message.	✓	✓*
`md5_hash`	MD5 hash of the affected file.	✓	✓
`sha256_hash`	SHA256 hash of the affected file.	✓	✓
`sha1_hash`	SHA1 hash of the affected file.	✓	✓
`sha3_384_hash`	SHA3-384 hash of the affected file.	✓	✓
`yara.rule_name`	Name of the matching YARA rule.	✓	✓*
`yara.author`	The author of the matching YARA rule.	✓	✓*
`yara.description`	Description of the matching YARA rule.	✓	✓
`yara.reference`	Reference of the matching YARA rule.	✓	✓
`yara.tlp`	Traffic Light Protocol (TLP) of the matching YARA rule.	✓	✓
Code Signing Certificate Blocklist (CSCB) additions			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓*
`type`	Type of this message.	✓	✗
`type`	Type of this message.	✓	✓*
`subject_cn`	Subject Common Name (CN).	✓	✓
`issuer_cn`	Subject Common Name (CN).	✓	✓
`algorithm`	Algorithm used.	✓	✓
`valid_from`	Datetime from when this Code Signing Certificate is valid from.	✓	✓
`valid_to`	Datetime to when this Code Signing Certificate is valid to.	✓	✓
`serial_number`	Serial number of the Code Signing Certificate.	✓	✓
`thumbprint_` `algorithm`	Thumbprint algorithm.	✓	✓
`thumbprint`	Thumbprint.	✓	✓
`bl_reason`	Code Signing Certificate Blocklist (CSCB) listing reason.	✓	✓*
`malware_samples`	List of malware samples signed with this Code Signing Certificate.	✓	✗

*** Comparable data for this field is provided via publicly available data, however the field names are not an exact match.**
If you require the comparable field name, please speak with your Spamhaus contact.

ThreatFox | Data exposed through Real Time Intelligence Feed v. publicly available data

ThreatFox provides context-rich signal from abuse.ch, sharing indicators of compromise (IOCs) associated with malware.

Users can access this data via the Real Time Intelligence Feed, or via publicly available APIs. These access methods provide different metadata outputs, as highlighted in this document.

The Real Time Intelligence Feed is the only access method to receive all observed threat signals.

Metadata field	Metadata description	Real Time Feed	Publicly available data
IOC additions			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓ *
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message. It's always 'ioc_addition'.	✓	✓
id	Is the ThreatFox ID of the IOC. You can also use this ID to craft the link to see the entry on the ThreatFox platform (https:// threatfox.abuse.ch/ioc/id/).	✓	✓
ioc	Is the IOC (value).	✓	✓
ioc_type	Is the type of the IOC (example: ip:port). A list of possible values is available through the API: https://threatfox.abuse.ch/api/#types	✓	✓
confidence_level	Is the confidence level of this IOC (set by the reporter). The value is between 0 and 100.	✓	✓
threat_type	Is the type of threat - a list of possible values is available through the API: https://threatfox.abuse.ch/api/#types	✓	✓
threat_type_ description	Is a short description, human-readable description, of threat_type.	✓	✓ *
malware	Is the malware family (using the Malpedia naming scheme).	✓	✓ *
malware_printable	Printable name of malware family (Malpedia)	✓	✓
malware_alias	Malware aliases (Malpedia)	✓	✓
sightings	Indicates how many times this IOC has been reported/observed.	✓	✗
anonymous	Boolean that indicates whether the submitter or this IOC wants to remain anonymous or not.	✓	✓
reporter	Is the abuse.ch handle of the submitter of this file (or 'null').	✓	✓
reward	List of rewards (credits) the reporter received from other users for this submission	✓	✓ *
tags	Is a List of tags associated with this file. A list of current tags is available through the API: https://threatfox.abuse.ch/api/#tag-list	✓	✓
reference	Reference (URL)	✓	✓
comment	Is a human-readable string comment from the reporter on this IOC.	✓	✓
IOC changes			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message. It's always 'ioc_change'.	✓	✗
id	Is the ThreatFox ID of the IOC. You can also use this ID to craft the link to see the entry on the ThreatFox platform (https:// threatfox.abuse.ch/ioc/id/).	✓	✗
ioc	Is The IOC (value).	✓	✗
ioc_type	This is the type of the IOC (example: ip:port). A list of possible values is available through the API: https://threatfox.abuse.ch/api/#types	✓	✗
threat_type	This is the threat type. A list of possible values is available through the API: https://threatfox.abuse.ch/api/#types	✓	✗
threat_type_ description	This is a short description, human-readable, of threat_type.	✓	✗
field	Shows the affected field where the change occurred.	✓	✗
value	Is the new value of the affected field.	✓	✗
action	Is an enumerated field that describes the action. May contain add, remove, change.	✓	✗
IOC removal			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message. It's always 'file_removal'.	✓	✗
id	Is the ThreatFox ID of the IOC. You can also use this ID to craft the link to see the entry on the ThreatFox platform (https:// threatfox.abuse.ch/ioc/id/).	✓	✗
ioc	Is the IOC (value).	✓	✗
ioc_type	This is the type of the IOC (example: ip:port). A list of possible values is available through the API: https://threatfox.abuse.ch/api/#types	✓	✗
threat_type	This is the threat type. A list of possible values is available through the API: https://threatfox.abuse.ch/api/#types	✓	✗
threat_type_ description	This is a short description, human-readable, of threat_type.	✓	✗
removal_note	Is a string containing any removal note.	✓	✗

*** Comparable data for this field is provided via publicly available data, however the field names are not an exact match.**
If you require the comparable field name, please speak with your Spamhaus contact.

YARAify | Data exposed through Real Time Intelligence Feed v. publicly available data

YARAify, provided by abuse.ch, is one of the largest repositories of YARA rules available. Users can scan suspicious files, such as malware samples or process dumps, against these rules to identify targeted attacks and threats, specific to their environment.

Users can access this data via the Real Time Intelligence Feed, or via publicly available APIs. These access methods provide different metadata outputs, as highlighted in this document.

The Real Time Intelligence Feed is the only access method to receive all observed threat signals.

Metadata field	Metadata description	Real Time Feed	Publicly available data
File Additions			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓ *
`uuid`	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✓ *
`type`	Defines the type of message. It's always 'file_addition'.	✓	✗
`md5_hash`	Is the MD5 hash of the payload received.	✓	✓
`sha256_hash`	Is the SHA256 hash of the payload received.	✓	✓
`sha1_hash`	Is the SHA1 hash of the file.	✓	✓
`sha3_384_hash`	Is the SHA3-384 hash of the file.	✓	✓
`file_size`	Is the size (in bytes) of the payload received.	✓	✓
`imphash`	Is the imphash of the payload received.	✓	✓
`ssdeep`	Is the ssdeep of the payload received.	✓	✓
`tlsh`	Is the tlsh of the payload received.	✓	✓
`telfhash`	Is the telfhash of the payload received.	✓	✓
`gimphash`	Is the gimphash of the file.	✓	✓
`dhash_icon`	Is the dhash of the file icon.	✓	✓
`mime_type`	Is the Multipurpose Internet Mail Extensions (MIME) type of the payload received.	✓	✓
Task results			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
`uuid`	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✓ *
`type`	Defines the type of message. It's always "task_result".	✓	✗
`task_id`	Task ID (UUID4).	✓	✓
`md5_hash`	Is the MD5 hash of the payload received.	✓	✓
`sha256_hash`	Is the SHA256 hash of the payload received.	✓	✓
`sha1_hash`	Is the SHA1 hash of the file.	✓	✓
`sha3_384_hash`	Is the SHA3-384 hash of the file.	✓	✓
`file_name`	Is the original file name.	✓	✓
`clamav_scan`	Boolean indicating whether the file has been scanned with ClamAV or not.	✓	✓
`unpack`	Is boolean value indicating whether the file has been processed by the Portable Executable (PE) unpacker.	✓	✓
`unpacked_files_cnt`	If `unpack` is `True`, number of unpacked files collected (if any).	✓	✓
`share_file`	Boolean indicating whether the user decided to share the sample or not.	✓	✓
`results.clamav`	Is the matching ClamAV signature.	✓	✓ *
`results.yara_static`	Is an array indicating the static YARA rule matching results.	✓	✓ *
`results.yara_unpack`	Is the array of the unpacker YARA rule matching results.	✓	✓ *
Unpacker results			
`_idx`	Is an integer representing the incremental number of the message.	✓	✗
`_ts`	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
`uuid`	UUID identifying this message	✓	✓
`type`	Type of this message	✓	✓
`md5_hash`	MD5 hash of the unpacked file	✓	✗
`sha256_hash`	SHA256 hash of the unpacked file	✓	✗
`sha1_hash`	SHA1 hash of the unpacked file	✓	✓
`sha3_384_hash`	SHA3-384 hash of the unpacked file	✓	✗
`file_name`	File name of the unpacked file	✓	✗
`file_size`	Size (in bytes) of the unpacked file	✓	✗
`timestamp`	Unix timestamp of the message	✓	✗
`imphash`	imphash of the unpacked file	✓	✗
`ssdeep`	ssdeep of the unpacked file	✓	✗
`tlsh`	TLSH of the unpacked file	✓	✗
`telfhash`	telfhash name of the unpacked file	✓	✗
`gimphash`	gimphash of the unpacked file	✓	✗
`dhash_icon`	dhash of the unpacked file' icon	✓	✗
`mime_type`	MIME type of the unpacked file	✓	✗
`parent_file`	The original file (parent) from which this file (child) got unpacked from	✓	✗
`yara_matches`	YARA rules matching this unpacked file	✓	✓

*** Comparable data for this field is provided via publicly available data, however the field names are not an exact match.**
If you require the comparable field name, please speak with your Spamhaus contact.

Feodo Tracker | Data exposed through Real Time Intelligence Feed v. publicly available data

Feodo Tracker rovides context-rich signal from abuse.ch, sharing botnet C&C infrastructure associated with major malware threats that facilitate ransomware attacks. This data helps network owners to protect their users.

Users can access this data via the Real Time Intelligence Feed, or via publicly available APIs. These access methods provide different metadata outputs, as highlighted in this document.

The Real Time Intelligence Feed is the only access method to receive all observed threat signals.

Metadata field	Metadata description	Real Time Feed	Publicly available data
Observed C2s			
_idx	Is an integer representing the incremental number of the message.	✓	✓
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✓
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✓
type	Defines the type of message. It's always 'observed_c2'.	✓	✓
ip_address	Is the IPv4 or IPv6 address of the botnet C2.	✓	✓ *
port	Is the port of the botnet C2.	✓	✓ *
protocol	Is the protocol the botnet C2 uses.	✓	✓ *
malware_malpedia	Is the malware family associated with this botnet C2 (using the Malpedia naming scheme).	✓	✓
as_number	Is the Autonomous System (AS) number associated with the botnet C2 (ip_address).	✓	✓
as_name	Is the AS name associated with the botnet C2.	✓	✓
country	Is the geo-located country of the botnet C2 (two-letter country code).	✓	✓
first_seen	Is the Unix timestamp when this botnet C2 has been observed for the first time.	✓	✓ *
first_seen	Is the Unix timestamp when this botnet C2 has been (re-)validated by Feodo Tracker last time.	✓	✓ *
last_online	Is the Unix timestamp when this botnet C2 has been seen active (online) for the last time.	✓	✓ *
C2 removal			
_idx	Is an integer representing the incremental number of the message.	✓	✗
_ts	Is the Unix timestamp, indicating when the message was received by the real time infrastructure.	✓	✗
uuid	Is an internal, unique identifier for the message. This is the property that should be used to dedupe the incoming flows where necessary.	✓	✗
type	Defines the type of message. It's always 'c2_removal'.	✓	✗
ip_address	Is the IPv4 of the botnet C2.	✓	✗
port	Is the port of the botnet C2.	✓	✗
protocol	Is the protocol the botnet C2 uses.	✓	✗
malware_malpedia	Is the malware family associated with this botnet C2 (using the Malpedia naming scheme).	✓	✗
removal_note	Contains the reason why the botnet C2 has been removed.	✓	✗

* **Comparable data for this field is provided via publicly available data, however the field names are not an exact match.**
If you require the comparable field name, please speak with your Spamhaus contact.