# Forensics API

The information and data accessible via this API contain Proofpoint proprietary, confidential, and/or trade secret information. Sharing or providing the information to another party without Proofpoint's express written consent is prohibited. Please review the updated Terms of Use for Proofpoint APIs. The TAP API Terms of Use can be found online at API Terms of Use | Proofpoint US.

# Overview

The Forensics API allows administrators to pull detailed forensic evidences about individual threats or campaigns observed in their environment. These evidences could be used as indicators of compromise to confirm infection on a host, as supplementary data to enrich and correlate against other security intelligence sources, or to orchestrate updates to security endpoints to prevent exposure and infection.

# API Features

## General Service Notes

1. All timestamps are in the returned events are in UTC.
2. Forensic results are returned in JSON format.
3. The results provided by this API may not be in any sorted order.

## Security

Each request:

- MUST use SSL.
- MUST use service credentials to authenticate to the API.
- MUST use the HTTP Basic Authorization method.
- MUST use the HTTP GET method

## Standard responses

Requests to the endpoints can produce a response with a variety of HTTP status codes. The following table describes the scenarios in which these codes can be produced.

| Code | Message | Scenarios |
|---|---|---|
| 200 | Success | At least one record matching the specified criteria was found and returned in the response body. In the case of JSON format, the structure is always returned, even if empty. |
| 400 | Bad Request | The request is missing a mandatory request parameter, a parameter contains data which is incorrectly formatted, or the API doesn't have enough information to determine the identity of the customer. |
| 401 | Unauthorized | There is no authorization information included in the request, the authorization information is incorrect, or the user is not authorized |
| 404 | Not Found | The campaign ID or threat ID does not exist. |
| 500 | Internal Server Error | The service has encountered an unexpected situation and is unable to give a better response to the request |

## Throttle Limits

To prevent the API from being overwhelmed by too many requests, the Forensics API throttles requests based on request types. Once exceeded, the API will start returning *429 HTTP* status codes until 24 hours past the oldest request has elapsed.

| Endpoint | Max Number o |
| --- | --- |
| /v2/forensics?threatId=<*threatId*>&includecampaignforensics=false | 50 calls per 24 hours per *threatId.* |
| /v2/forensics?threatId=<*threatId*>&includecampaignforensics=true | 1800 per 24 hours. |
| /v2/forensics?campaign=<*campaignId*> | Different t*hreatId* and campaignId ar request quota. |

## Available Endpoints

All endpoints are available on the *tap-api-v2.proofpoint.com* host. For example, *https://tap-api-v2.proofpoint.com/v2/forensics*.

#### /v2/forensics

Fetch forensic information for a given threat or campaign.

## Required Parameters

One of the following two parameters MUST be supplied with each request. They are mutually exclusive.

#### threatId

An string containing a threat identifier.

#### campaignId

An string containing a campaign identifier.

## Optional Parameters

One or more of these parameters may also be provided.

**includeCampaignForensics**

A boolean value, defaulting to false. May optionally be used with the threatId parameter. It cannot be used with the campaignId parameter.

If false, aggregate forensics for that specific threat identifier will be returned.

If true AND if the threat has been associated with a campaign, aggregate forensics for the entire campaign are returned. Otherwise, aggregate forensics for the individual threat are returned.

## Example Commands in Curl

The following commands assume that PRINCIPAL and SECRET are defined environment variables. They correspond to the service principal and secret that was created on the Settings page.

> *curl "https://tap-api-v2.proofpoint.com/v2/forensics?threatId=<threatId>" --user "$PRINCIPAL:$SECRET" -s*

Will return aggregate forensics for the specified threat only.

> *curl "https://tap-api-v2.proofpoint.com/v2/forensics?threatId=<threatId>&includeCampaignForensics=true" --user "$PRINCIPAL:$SECRET" -s*

Returns aggregate forensics for an entire campaign, if the threat has been associated with a campaign.

> *curl "https://tap-api-v2.proofpoint.com/v2/forensics?campaignId=<campaignId>" --user "$PRINCIPAL:$SECRET" -s*

Returns aggregate forensics for an entire campaign.

# Aggregate Forensics Format

The aggregate forensics format is a JSON structure that contains nested objects.

## Top-level Structure

The top-level structure contains the time the report was generated and an array of independent forensics reports for one or more threat variants.

```
{
   "generated" : "string", //ISO8601-formatted datetime corresponding to the time this report was generated
   "reports" : [ list<Report> ] //array of Report objects
}
```

## Report Objects

Each report object identifies which threat or campaign the report is associated with, it's type, and report scope. Report

objects contain one or more Evidence objects.

```
{
   "name" : "string", //the malicious URL, SHA256 hash of the malicious attachment, or campaign name
   "scope" : "string", //whether the report scope covers a campaign or an individual threat
   "type" : "string", // the threat type: attachment, url, or hybrid
   "id" : "string", // the identifier associated with the campaign or individual threat
   "forensics" : [ list<Evidence> ] //array of Evidence objects
}
```

## Evidence Objects

Evidence objects correspond to markers that were detected by Proofpoint's sandboxes or indicators of compromise which were produced by Proofpoint's team of threat analysts. Each evidence has a type, maliciousness, details, and an array of Platform objects on which the evidence was observed.

```
{
   "type" : "string", // the evidence type
   "display" : "string", // a friendly display string
   "malicious" : "string", // whether the evidence was used to reach a malicious verdict
   "time" : "string", // [Unsupported] This field is currently unsupported and it's expected to always return 0.
   "what" : { map<EvidenceTypes> }, // a map of values associated with the specific evidence type, see
Evidence Types section below
   "platforms" : [ list<Platform> ] //array of Platform objects
}
```

## Platform Objects

Each Platform object corresponds to an environment in which its parent Evidence object was seen. Examples of platforms include Windows XP, Windows 7, or Android.

```
{
   "name" : "string",
   "os" : "string",
   "version" : "string"
}
```

# Evidence Types

There are many different evidence types that can be produced by the sandboxes. Each evidence type can have different facets associated with it within the "what" attribute.

## Attachment

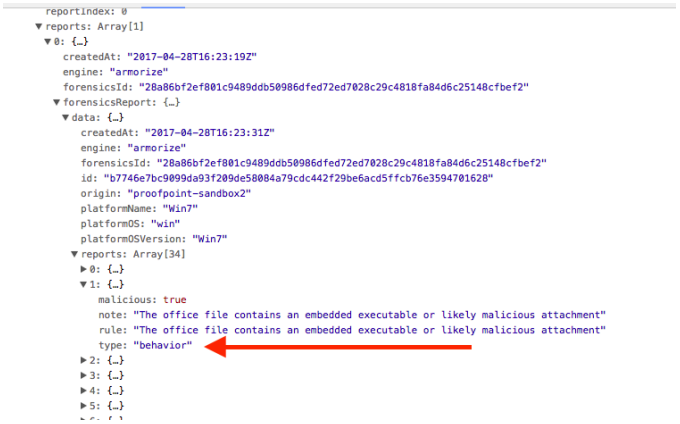
An email attachment was scanned. The attachments are usually matched against a static rule which has identified some

portion of their content as objectionable. Objectionable content might include malicious macros, known phishing templates, or exploits against the filetype's structure. The presence of an attachment evidence is a strong indicator that the attachment is malicious.

```
"type" : "attachment",
"what" : {
    "sha256" : "string", // the SHA256 hash of the attachment's contents
    "blacklisted" : integer, // optional, whether the file was blocklisted
    "md5" : "string", // optional, the MD5 sum of the attachment's contents
    "offset" : integer, // optional, the offset in bytes where the malicious content was found
    "rule" : "string" // optional, the name of the static rule inside the sandbox which located the malicious content
    "size" : integer // optional, the size in bytes of the attachment's contents
}
```

## Behavior

A malicious behavior was observed. The malicious "behavior" types are effectively algorithms that aim to match some percentage of the scans obtained from the sandbox. The presence of a behavior evidence indicates that the behavior is very likely malicious. When a behavior evidence is definitively identified, its signature is available. Malicious content may include malicious macros, spyware, and Trojan horses, among other malware types.




## Cookie

Cookies were set or deleted in the sandbox's browser as a result of the scan. Among other things, cookies can be used by attackers to recognize previously victimized users and hiding data used by other processes. However, there are many legitimate uses of cookies, so the presence of cookie activity is not itself an indicator of maliciousness. Combined with other observations, it may be a useful indicator.

```
"type" : "cookie",
"what" : {
    "action" : "string", // whether the cookie was set or deleted
    "domain" : "string", // which domain set the cookie
    "key" : "string", // the name of the cookie being set or deleted
    "value" : "string" // optional, content of the cookie being set
}
```

## DNS

The sandbox resolved a hostname as a result of the scan. While there is nothing inherently malicious about DNS lookups being performed, they can be useful indicators when combined with other evidences. In many cases, attackers use disposable hostnames which point to longer-lived CNAMEs or nameserver infrastructure,

```
"type" : "dns",
"what" : {
   "host" : "string", // the hostname being resolved
   "cnames" : [ list<string> ], // optional, an array of cnames which were associated with the hostname
   "ips" : [ list<string> ], // optional, an array of IP addresses which were resolved to the hostname
   "nameservers" : [ list<string> ], // optional, the nameservers responsible for the hostname's domain
   "nameserversList" : [ list<string> ] // optional, the nameservers responsible for the hostname's domain
}
```

## Dropper

A dropper was observed in the sandbox as a result of the scan. A dropper is usually a simple stub that's designed to retrieve a malware payload from attacker infrastructure. Dropper evidences are positive indicators of maliciousness.

```
"type" : "dropper",
"what" : {
   "path" : "string", // the location of the dropper file
   "rule" : "string", // optional, the name of the static rule inside the sandbox which identified the dropper
   "url" : "string" // optional, the URL the dropper contacted
}
```

## File

Filesystem activity was observed in the sandbox as a result of the scan. Files are modified by many different processes, but strong indicators of maliciousness include overwriting system files, creating look-alike system files, attempting to establish persistence by writing to the Startup folders, etc.

```
"type" : "file",
"what" : {
   "action" : "string", // optional, the filesystem call made: create, modify, or delete
   "md5" : "string", // optional, the MD5 sum of the file's contents
   "path" : "string", // optional, the location of the file operated on
   "rule" : "string", // optional, the name of the static rule inside the sandbox which identified the suspicious file
   "sha256" : "string", // optional,the SHA256 hash of the file's contents
   "size" : integer // optional, the size in bytes of the file's contents
}
```

## IDS

An IDS rule was triggered from the sandboxes' host as a result of the scan. The presence of an IDS trigger is an extremely strong confirmation of malicious intent.

```
"type" : "ids",
"what" : {
   "name" : "string", // the friendly name of the IDS rule which observed the malicious traffic
   "signatureId" : integer // the identifier of the IDS rule which observed the malicious traffic
}
```

## Mutex

A mutex was created in the sandbox's OS as a result of the scan. Mutexes are created by processes looking to synchronize threads within a process, or for IPC. Attackers commonly use mutexes to prevent multiple instances of a malcious payload from infecting the same host. The presence of mutex evidences are a strong indicator of maliciousness.

```
"type" : "mutex",
"what" : {
   "name" : "string", // the name of the mutex
   "path" : "string" // optional, the path to the process which spawned the mutex
}
```

## Network

Network activity on the sandbox was initated as a result of the scan. This isn't itself evidence of maliciousness, but combined with other evidences it can be a strong signal. Examples of malicious network activity might include contacting attacker command-and-control infrastructure, listening on local ports to provide remote access, or scanning the local network to spread infection horizontally.

```
"type" : "network",
"what" : {
   "action" : "string", // the type of network activity being initated: connect or listen
   "ip" : "string", // the remote IP address being contacted
   "port" : "string", // the remote IP port being contacted
   "type" : "string", // the protocol being used: tcp or udp
}
```

## Process

A process was created on the sandbox was initated as a result of the scan. A process which was launched in response to an attachment being opened or a URL being visited is usually a strong indicator of malicious intent.

```
"type" : "process",
```

```
"what" : {
   "path" : "string", // the location of the executable which spawned the process
   "action" : "string" // the action peformed on the process, current only create is produced
}
```

## Registry

Registry modifications were made on the sandbox as a result. Registry modifications can be innocuous, but attempts to establish process persistence or disable security software can be identified in these evidences in many cases.

```
"type" : "registry",
"what" : {
   "action" : "string", // the registry change made: create or set
   "key" : "string", // the location of the registry key being modified
   "name" : "string", // optional, the name of the registry entry being created or set
   "rule" : "string", // optional, the name of the static rule inside the sandbox which identified the suspicious
registry modification
   "value" : "string" // optional, the contents of the key being created or set
}
```

## Screenshot

An automated screenshot was captured by the sandbox during the scan.

```
"type" : "screenshot",
"what" : {
   "url" : "string", // the URL hosting the of the screenshot image
}
```

## URL

A URL was visited by the sandbox as a result of the scan. There is nothing necessarily malicious about a URL being visited, but combined with other evidences, it can be a strong indicator of maliciousness. If the URL is blocklisted, Proofpoint knows it to be part of a campaign or attacker infrastructure. Rules can indicate the presence of objectionable content, such as phishing templates, exploit kit landing pages, or attempts to perform advanced victim selection.

```
"type" : "url",
"what" : {
   "url" : "string", // the URL which was observed
   "blacklisted", : boolean, // optional, whether the URL was listed on a blocklist
   "ip" : "string", // optional, the IP address that was resolved to the hostname by the sandbox
   "httpStatus" : integer, // optional, the HTTP status code which was produced when our sandbox visited the
URL
   "md5" : "string", // optional, the md5 value of the file downloaded from the URL
   "offset" : integer, // optional, the offset in bytes where the URL was found
   "rule" : "string", // optional, the name of the static rule inside the sandbox which located the URL
```

```
    "sha256" : "string", // optional, the sha256 value of the file downloaded from the URL
    "size" : integer // optional, the size in bytes of the file retrieved from the URL
}
```