Check Point
SOFTWARE TECHNOLOGIES LTD.
WELCOME TO THE FUTURE OF CYBER SECURITY

THREAT PREVENTION API

QUICK AND EASY DEMO GUIDE

Purpose

This document outlines some Threat Prevention API use cases and commands that SEs can run to demonstrate to customers the use of the API, and see some example responses.

The commands are designed to rely on an absolute minimum of infrastructure – all you need is a cloud API key and a standard Check Point SE laptop. The commands are run in Windows PowerShell using `curl`. They will also work from a Linux client.

Feel free to give the commands in this document to customers to assist them in implementing their API client application

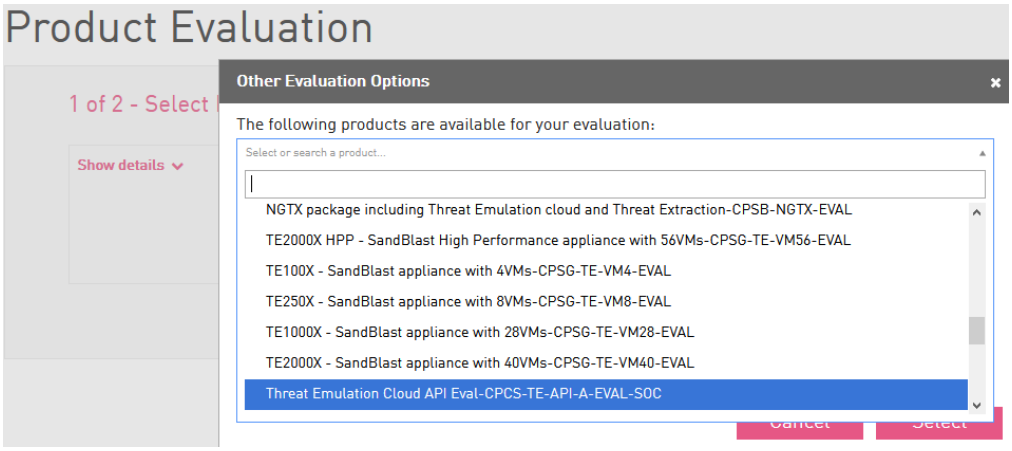
REQUIREMENTS

You will need to have a file that you can test with.

In order to show the full set of responses, this file should be one that ThreatCloud has never seen before. You can do this by creating a new document. Place it somewhere near the top level directories on your hard disk because you will need to type the whole path, so shorter is easier.

You will also need a cloud API key.
This can be generated from UserCenter/PartnerMap in the usual way. From the top menu, navigate to Product Evaluations and select Other Evaluation Option at the right.

Search down the list until you find the Threat Emulation section, and select the API cloud eval as shown below.



COMMANDS

First, it is important to know that Windows PowerShell doesn't implement `curl` as we know it from Check Point or *nix environments. In PowerShell, `curl` is aliased to `Invoke-WebRequest`, which doesn't use the same syntax, so we must remove this alias before we can use the cross-platform commands in this document.

Remove the alias as follows:

```
PS C:\Users\andyn> Remove-item alias:curl
```

This will remove the alias for this PowerShell session only.

UPLOAD COMMAND

To upload your file, we must send an API call to the Threat Emulation Cloud Scanning service.

The command contains the following parts:

1. `curl`

   The program we're running.

2. `--request POST 'https://te.checkpoint.com/tecloud/api/v1/file/upload'`

   The type of request (POST) and the URL we're sending it to.

3. `--header 'Authorization:TE_API_KEY_<YOUR API KEY GOES HERE>'`

   Your API Key that you generated when you obtained the evaluation in the section above.

4. `--form 'file=@<FULL PATH TO FILE GOES HERE>'`

   The full path to the file. Note that you MUST escape any spaces in the filename with a \ character.

5. `--form 'request={ "request": [ { "features": [ "te" ], "te": { "reports": [ "xml", "summary" ], "images": [ { "id": "e50e99f3-5963-4573-af9e-e3f4750b55e2", "revision": "1" } ] } } ]}'`

   The options for the request. In this example,

   Use the TE feature.

   For the TE feature, please produce XML and Summary reports (XML reports could be downloaded and parsed by a system to get more information about the file after the emulation is complete).

   Analyse the file in the image with ID starting `e50`, which happens to be Windows XP. The IDs for other images are listed in the appendix of this document.

   Use this revision of the image. This can be any number, in fact, but it's a required parameter. The scanning centre will use the latest version if you leave this to "1".

Putting this together, we get the following command which you can copy, edit in the API key and file name and run.

```
curl --request POST 'https://te.checkpoint.com/tecloud/api/v1/file/upload' --header
'Authorization: TE_API_KEY_<YOUR API KEY GOES HERE>' --form 'file=@<FULL PATH TO FILE>'
--form 'request={ "request": [{"features": ["te"],"te":{"reports":["xml","summary"],
"images":[{"id":"e50e99f3-5963-4573-af9e-e3f4750b55e2","revision":"1"}]}}]}'
```

Replace all the text with yellow highlighting, including the <> characters.

UPLOAD EXAMPLE

Here is an example command and the output you should see:

```
PS C:\Users\andyn> curl --request POST 'https://te.checkpoint.com/tecloud/api/v1/file/upload' --header 'Authorization: T
E_API_KEY [redacted]' --form 'file=@d:\SmallTest6.docx' --form 'request={  "request": [ {
"features": [ "te" ], "te": { "reports": [ "xml", "summary" ], "images": [ { "id": "e50e99f3-5963-4573-af9e-e3f4750b55e2
", "revision": "1" } ] } } ]}'
{
  "response": {
    "status": {
      "code": 1002,
      "label": "UPLOAD_SUCCESS",
      "message": "The file was uploaded successfully."
    },
    "sha1": "695ddc997df998369bc8c875d3095ce593d681a6",
    "md5": "cba90edcc9ddd02379934797eb110a04",
    "sha256": "dff04b72806da4ee8d46163182d719fdca6c674c8015847abc33f7b7729aab00",
    "file_type": "",
    "file_name": "SmallTest6.docx",
    "features": [
      "te"
    ],
    "te": {
      "trust": 0,
      "images": [
        {
          "report": {
            "verdict": "unknown"
          },
          "status": "not_found",
          "id": "e50e99f3-5963-4573-af9e-e3f4750b55e2",
          "revision": 1
        }
      ],
      "score": -2147483648,
      "status": {
        "code": 1002,
        "label": "UPLOAD_SUCCESS",
        "message": "The file was uploaded successfully."
      }
    }
  }
}
PS C:\Users\andyn>
```

QUERY COMMAND

Once you have uploaded the file, copy one of the hashes from the output to your clipboard so that you can then query for the result.

To query for the verdict of a file, we send a slightly different API call

1. `curl`

   Same command as before.

2. `--request POST 'https://te.checkpoint.com/tecloud/api/v1/file/query'`

   Again, type of request and URL. Notice the different ending on the URL.

3. `--header 'Authorization: TE_API_KEY_<YOUR API KEY GOES HERE>'`

   Same header with API key.

4. `--header 'Content-Type: application/json'`

   As we're not uploading a file, we need a different header to declare that we're using JSON.

5. `--data-raw '{"request" :{"sha256":"<HASH HERE>","features":["te"],"te":{"images":[{"id":"e50e99f3-5963-4573-af9e-e3f4750b55e2","revision":1}]}}}'`

   The request itself. Notice we now declare which hash type we're using and include the hash value.

Again, putting this together we can get a command you can run.

```
curl --location --request POST 'https://te.checkpoint.com/tecloud/api/v1/file/query' --header 'Authorization: TE_API_KEY_<APIKEY>' --header 'Content-Type: application/json' --data-raw '{"request": {"sha256":"<HASH>","features":["te"],"te":{"images":[{"id": "e50e99f3-5963-4573-af9e-e3f4750b55e2","revision":1}]}}}'
```

QUERY EXAMPLE

And as before, here is an example of the output you should see:

```
PS C:\Users\andyn> curl --location --request POST 'https://te.checkpoint.com/tecloud/api/v1/file/query' --header 'Author
ization: TE_API_KEY [redacted]' --header 'Content-Type: application/json' --data-raw '{"re
quest" :{"sha256":"dff04b72806da4ee8d46163182d719fdca6c674c8015847abc33f7b7729aab00","features":["te"],"te": {"images":[
{"id": "e50e99f3-5963-4573-af9e-e3f4750b55e2","revision":1}]}}}'
{
  "response": {
    "status": {
      "code": 1001,
      "label": "FOUND",
      "message": "The request has been fully answered."
    },
    "sha256": "dff04b72806da4ee8d46163182d719fdca6c674c8015847abc33f7b7729aab00",
    "file_type": "docx",
    "file_name": "",
    "features": [
      "te"
    ],
    "te": {
      "trust": 0,
      "images": [
        {
          "report": {
            "verdict": "benign"
          },
          "status": "found",
          "id": "e50e99f3-5963-4573-af9e-e3f4750b55e2",
          "revision": 1
        }
      ],
      "score": -2147483648,
      "combined_verdict": "benign",
      "status": {
        "code": 1001,
        "label": "FOUND",
        "message": "The request has been fully answered."
      }
    }
  }
}
PS C:\Users\andyn>
```

The important parts to note in the response are:

```
"score": -2147483648,
      "combined_verdict": "benign",
      "status": {
      "code": 1001,
      "label": "FOUND",
      "message": "The request has been fully answered."
```

Note that if you include multiple images you will get a response section for each of them, to confirm which OS versions of the Threat Emulation sandbox found the file to be malicious or benign.

APPENDIX

OS Image IDs for emulating files in different OS versions:

Note that these are case sensitive!

WinXP, Office 2003/7	e50e99f3-5963-4573-af9e-e3f4750b55e2
Win7, Office 2003/7	7e6fe36e-889e-4c25-8704-56378f0830df
Win7, Office 2010	8d188031-1010-4466-828b-0cd13d4303ff
Win7, Office 2013	5e5de275-a103-4f67-b55b-47532918fa59
Win7 64bit,Office 2010	3ff3ddae-e7fd-4969-818c-d5f1a2be336d
Win8.1 64bit,Office 2013	6c453c9b-20f7-471a-956c-3198a868dc92
Win10, 74bitOffice 2016	10b4a9c6-e414-425c-ae8b-fe4dd7b25244

Further Reading:

Threat Prevention API 1.0 Reference Guide: link