CAWS CONTINUOUS SECURITY VALIDATION PLATFORM API GUIDE

VERSION 3.4.0

NSS LABS

Version 3.4, 12/21/2017

NSS Labs, Inc.
206 Wild Basin Road
Building A, Suite 200
Austin, TX 78746 US
info@nsslabs.com
www.nsslabs.com

Introduction to GraphQL

GraphQL is a query language for your API, and a server-side runtime for executing queries by using a type system you define for your data. GraphQL is not tied to any specific database or storage engine and is instead backed by your existing code and data.

Type System

The GraphQL query language is about selecting fields on objects. So, for example, in the following query:

Example GraphQL Query Request

```
{
    threatAnalytics {
        applicationThreatCounts {
            count
        }
    }
}
```

1. Start with a special "root" object.
2. Select the **threatAnalytics** field.
3. For the object returned by **hero**, select the **applicationThreats** field, which has a **count** field.

Because the shape of a GraphQL query closely matches the result, you can predict what the query will return without knowing that much about the server. It is useful to have an exact description of the data we can ask for - what fields can we select? What kinds of objects might they return? What fields are available on those sub-objects? That's where the schema comes in. Every GraphQL service defines a set of types which completely describe the set of possible data you can query on that service. Then, when queries come in, they are validated and executed against that schema.

Object and Field Types

The most basic components of a GraphQL schema are object types, which represent a kind of object you can fetch from a service, and what fields it has. In the GraphQL schema language, we might represent it like this:

Example GraphQL Type

```
type ExploitInfo {
    nssid: ID
    discoveryDate: String!
    type: [ExploitType]!
    processCount: Int
    maliciousProcessCount: Int
    relatedExploits: Int
    fileCount: Int
    connectionCount: Int
}
```

- **ExploitInfo** is a GraphQL Object Type, meaning it's a type with some fields. Most of the types in your schema will be object types
- **nssid** through **connectionCount** are fields on the **ExploitInfo** type. That means that they are the only fields that can appear in any part of a GraphQL query that operates on the **ExploitInfo** type
- **String** is one of the built-in scalar types. These are types that resolve to a single scalar object, and can't have sub-selections in the query

- **String!** means that the field is non-nullable, meaning that the GraphQL service promises to always give you a value when you query this field. In the type language, we'll represent those with an exclamation mark
- **[ExploitType]!** represents an array of **ExploitType** objects. Since it is also non-nullable, you can always expect an array (with zero or more items) when you query the **type** field

Query and Mutation Types

Most types in a schema will just be normal object types, but there are two types that are special within a schema. Every service has a query type and may or may not have a mutation type. A query is a type that defines a data set that can be selected. A mutation is a type that defines an operation that creates or updates a data set. Here is an example of a query type called threatInfo that takes two parameters:

Example GraphQL Query

```
{
    threatInfo(start: String, end: String) {
        exploits {
            nssid,
            discoveryDate,
            type,
            processCount,
            relatedExploits
        }
    }
}
```

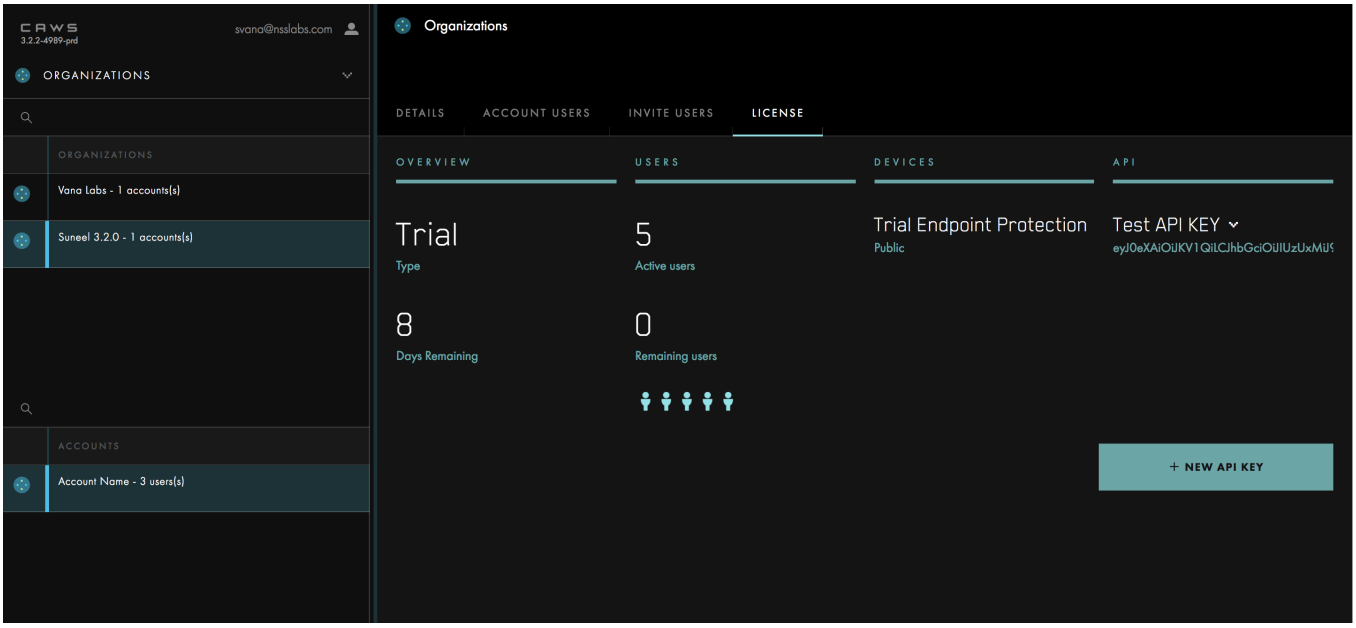
API TOKEN

Every request sent to the CAWS 3 API requires an authorization header with a API Token as a value. This allows the API to ensure you are authenticated and allowed to send the request. The API Token can be generated by logging to CAWS UI and navigating to the Organziations -> License section.

How to generate the API Key:

- Login to CAWS UI https://caws3.nsslabs.com using your credentials
- Naviate to ORGANIZATIONS section by selecting ORGANIZATIONS from drop down menu
- Click on LICENSE to navigate to license tab
- Click on NEW API KEY button to genereate a API Key
- Copy and save the API Key and use it with all the API Calls as defined in the examples

The screenshot below shows an example API KEY created using CAWS UI.



Example API Key:

```
1. MWM1LWVlNTQtNGUE_5_PoQHr1SIF1JBNd72s93q4Q28BMIe - hf62lpgw.....................
```

The API Key has been truncated for the convenience of documentation.

ACCOUNT ID

This query allows you get a collection of all organizations and accounts you have access to. In CAWS, the entitlement model allows a single organization to have multiple child accounts. And as a user, you may have access to multiple accounts. An account is the entry point to all operations you can perform using the API. As such, most API calls require an account that the API operation will be performed against. For example, to query for a collection of profiles, you need to provide the account whose profiles you want to query. This is a new query that has no equivalent in the CAWS 2 API.

Parameters

None

Request and Response Structure

See the online documentation(https://caws3.nsslabs.com/docs/securityprofileoverview.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Curl Example API Call to Get Account ID

```
curl  - X  POST  https: //caws3.nsslabs.com/graphql -
d query='query { accounts { id name organization { id name } } }' -
H 'authorization: bearer <API Token>'
```

Example Output for Get Token Query

```
{
    "data": {
        "accounts": [{
            "id": "1",
            "name": "Account 1",
            "organization": {
                "id": "1",
                "name": "NSS 1"
            }
        }, {
            "id": "2",
            "name": "Account 2",
            "organization": {
                "id": "2",
                "name": "NSS 2"
            }
        }]
    }
}
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests
```

```
# ...
#
# Function to Get Your Organization and Account(s) Information.
def get_organization(token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            query Organizations{
                organizations{
                    id
                     name
                    accounts{
                             id,
                             name
                    }
                }
            }
            """

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }

    # Define Request Data to Send to Server.
    #
    # This is both the query and any variables in requires. Remember, GraphQL
    # is JSON based so the variables parameter in the request has to be a JSON
    # string!
    data = {
        'query': query
    }

    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
```

```
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('organizations')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

CREATE SECURITY PROFILE

This mutation allows you to create a new profile. This is a new mutation that has no equivalent in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1
name	string	The name to associate with the profile. This is used for future reference
description	string	A description to associate with the profile
applications	int[], array of integers	An array of application ids you want to associate with the profile. Example: [1,2,3]
device	int	A platform id you want to associate with the profile. Example: 1
platforms	int[], array of integers	An array of platform ids you want to associate with the profile. Example: [1,2,3]
location	string	An optional location to associate with the profile
notificatiionEnabled	Boolean	Flag to enable/disable notifications for a profile.

Request and Response Structures

See the online documentation(https://caws3.nsslabs.com/docs/securityprofilesummary.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in the request JSON structure will be returned in the response JSON structure.

```
import json
import requests

def create_profile(account, name, description, device, applications, platforms,
token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
```

```
# sending a GraphQL Query.
#
# Remember, GraphQL is graph based and we have to explicitly specify the
# fields that we want returned in the response.
#
# For an online reference to all the different GraphQL Queries and
# Mutations supported by CAWS, including their entire response data
# structures, the online documentation at https://caws3.nsslabs.com/docs
# is your best friend.
query = """
        mutations CreateSecurityProfile
           (
                  $account: ID!, $name: String!, $description: String!,
                  $applications: [ID!]!,
                  $notiificationEnabled: $notiricationEnabled,
                  $device: ID!, $platforms: [ID!]!
           )
           {
                  createSecurityProfile
           (
           account: $account, name: $name, description: $description,
           applications: $applications,
           notificationEnabled: $notificationEnabled, device: $device,
           platforms: $platforms
           )
           {
                id
                  }
           }
        """

# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'account': account,
    'name': account,
    'description': description,
    'applications': applications,
    'device': device,
    'platforms': platforms
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
# call to the "authenticateUser" Mutation. See this manual for an example
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
```

```
    }

    # Define Request Data to Send to Server.
    #
    # This is both the query and any variables in requires. Remember, GraphQL
    # is JSON based so the variables parameter in the request has to be a JSON
    # string!
    data = {
        'query': query,
        'variables': json.dumps(variables)
    }

    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('createSecurityProfile')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

GET SECURITY PROFILES

This query allows you to retrieve a list of all the profiles associated with your account. You will primarily need this query to get a list of your profile ids so that you can further use them to query for details pertaining to each profile. This is a new query that has no equivalent in the CAWS 2 API.

Here is an CURL example to get profile IDs.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1

Request and Response Structure

See the online documentation (https://caws3.nsslabs.com/docs/securityprofileoverview.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Get Security Profiles Curl API Call Example

```
curl -X POST https://caws3.nsslabs.com/graphql -
d query='query { securityProfiles(account: "1") { id name }}' -H 'authorization: bearer <API Token>'
```

Example Output for Get Profile ID

```
{
    "data": {
        "securityProfiles": [{
            "id": "8",
            "name": "Profile 8"
        }, {
            "id": "9",
            "name": "Profile 9"
        }]
    }
}
```

Get Security Profiles Python API Call Example:

```
import json
import requests

# ...
#
# Function to Get a List of Profiles.
def get_profiles(account, token):
    # Define Query to Send to Server.
```

```
#
# The query that is sent to the server is either a GraphQL Query or a
# GraphQL Mutation. In this example because we want to get data, a we are
# sending a GraphQL Query.
#
# Remember, GraphQL is graph based and we have to explicitly specify the
# fields that we want returned in the response.
#
# For an online reference to all the different GraphQL Queries and
# Mutations supported by CAWS, including their entire response data
# structures, the online documentation at https://caws3.nsslabs.com/docs
# is your best friend.
query = """
        query SecurityProfiles($account: ID!){
            securityProfiles(account: $account){
                id
                name
            }
        }

        """
# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'account': account
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
# call to the "authenticateUser" Mutation. See this manual for an example
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
}

# Define Request Data to Send to Server.
#
# This is both the query and any variables in requires. Remember, GraphQL
# is JSON based so the variables parameter in the request has to be a JSON
# string!
data = {
    'query': query,
    'variables': json.dumps(variables)
}

# Send Request to Server.
#
# Unlike traditional REST APIs, in GraphQL, there is always a single
```

```
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('securityProfiles')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

GET GLOBAL THREAT ANALYTICS

This query allows you to get global analytics, not tied to a specific profile, pertaining to drive-by and file exploits that have been detected by CAWS. This is a new query that has no equivalent in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
start	string	The start date you want to retrieve data from, in ISO-8601 format. Example: "2017-03-01"
end	string	The end date you want to retrieve data to, in ISO-8601 format. Example: "2017-04-01"
applications	int[], array of integers	An optional array of application ids you want to restrict data for. Example: [1,2,3]
platforms	int[], array of integers	An optional array of platform ids you want to restrict data for. Example: [1,2,3]
countries	string[], array of strings	An optional array of country codes, in ISO-3166 format, you want to restrict data for. Example: ["US"]

Request and Response Structure

See the online documentation (https://caws3.nsslabs.com/docs/securityprofileoverview.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in the request JSON structure will be returned in the response JSON structure.

Example Curl API call to get Global Threat Analytics for given date range

```
curl https://caws3.nsslabs.com/graphql -d 'query=query {
        threatAnalytics(start:  "2017-06-17", end:  "2017-06-17") {
        applicationThreatCounts {
            application {
                id         name         family         vendor {
                    id           name
                }
            }
            count
        }
        countryThreatCounts {
            countryCode       count
        }
        platformThreatCounts {
            platform {
                id         name         family         vendor {
                    id           name
                }
            }
        }
        metrics {
```

```
            exploits {
                count
            }
            scannedFiles {
                count
            }
            scannedUrls {
                count
            }
        }
    }
}
' 'authorization: bearer <API Token>'
```

JSON Response


```
{
    "data": {
        "threatAnalytics": {
            "applicationThreatCounts": [{
                "application": {
                    "id": "176",
                    "name": "Adobe Flash Player 18.0.0.160",
                    "family": "Adobe Flash Player",
                    "vendor": {
                        "id": "1",
                        "name": "Adobe"
                    }
                },
                "count": 2
            }, {
                "application": {
                    "id": "178",
                    "name": "Adobe Flash Player 18.0.0.268",
                    "family": "Adobe Flash Player",
                    "vendor": {
                        "id": "1",
                        "name": "Adobe"
                    }
                },
                "count": 1
            }],
            "countryThreatCounts": [{
                "countryCode": "US",
                "count": 19
            }],
            "platformThreatCounts": [{
                "platform": {
                    "id": "2",
                    "name": "Windows 7 SP1",
                    "family": "Windows",
                    "vendor": {
                        "id": "18",
                        "name": "Microsoft"
                    }
                }
            }, {
                "platform": {
                    "id": "4",
                    "name": "Windows 8.1",
                    "family": "Windows",
```

```
                    "vendor": {
                        "id": "18",
                        "name": "Microsoft"
                    }
                }
            }],
            "metrics": {
                "exploits": {
                    "count": 19
                },
                "scannedFiles": {
                    "count": 0
                },
                "scannedUrls": {
                    "count": 143800
                }
            }
        }
    }
}
```

Example API call to get Global Threat Analytics for given date range and application

```
curl https://caws3.nsslabs.com/graphql -d 'query=query {
        threatAnalytics(start:  "2017-06-17", end:  "2017-06-17", applications: [176]) {
        applicationThreatCounts {
            application {
                id         name         family         vendor {
                    id           name
                }
            }
            count
        }
        countryThreatCounts {
            countryCode       count
        }
        platformThreatCounts {
            platform {
                id         name         family         vendor {
                    id           name
                }
            }
        }
        metrics {
            exploits {
                count
            }
            scannedFiles {
                count
            }
            scannedUrls {
                count
            }
        }
    }
}  ' -H 'authorization: bearer <API Token>'
```

JSON Response


```
{
    "data": {
        "threatAnalytics": {
            "applicationThreatCounts": [{
                "application": {
                    "id": "176",
                    "name": "Adobe Flash Player 18.0.0.160",
                    "family": "Adobe Flash Player",
                    "vendor": {
                        "id": "1",
                        "name": "Adobe"
                    }
                },
                "count": 2
            }],
            "countryThreatCounts": [{
                "countryCode": "US",
                "count": 2
            }],
            "platformThreatCounts": [{
                "platform": {
                    "id": "2",
                    "name": "Windows 7 SP1",
                    "family": "Windows",
                    "vendor": {
                        "id": "18",
                        "name": "Microsoft"
                    }
                }
            }],
            "metrics": {
                "exploits": {
                    "count": 2
                },
                "scannedFiles": {
                    "count": 0
                },
                "scannedUrls": {
                    "count": 145391
                }
            }
        }
    }
}
```


Example API call to get Global Threat Analytics for given date range, Application, Platform and Country

```
curl https://caws3.nsslabs.com/graphql -d 'query=query {
        threatAnalytics(start:  "2017-06-17", end:  "2017-06-
17", applications: [176], platforms: [2], countries: ["US"]) {
        applicationThreatCounts {
            application {
                id         name         family         vendor {
```

```
                    id          name
                }
            }
            count
        }
        countryThreatCounts {
            countryCode       count
        }
        platformThreatCounts {
            platform {
                id          name          family          vendor {
                    id          name
                }
            }
        }
        metrics {
            exploits {
                count
            }
            scannedFiles {
                count
            }
            scannedUrls {
                count
            }
        }
    }
}  ' -H 'authorization: bearer <API Token>'
```

JSON Response

```
{
    "data": {
        "threatAnalytics": {
            "applicationThreatCounts": [{
                "application": {
                    "id": "176",
                    "name": "Adobe Flash Player 18.0.0.160",
                    "family": "Adobe Flash Player",
                    "vendor": {
                        "id": "1",
                        "name": "Adobe"
                    }
                },
                "count": 2
            }],
            "countryThreatCounts": [{
                "countryCode": "US",
                "count": 2
            }],
            "platformThreatCounts": [{
                "platform": {
                    "id": "2",
                    "name": "Windows 7 SP1",
                    "family": "Windows",
                    "vendor": {
                        "id": "18",
                        "name": "Microsoft"
                    }
```

```
                }
            }],
            "metrics": {
                "exploits": {
                    "count": 2
                },
                "scannedFiles": {
                    "count": 0
                },
                "scannedUrls": {
                    "count": 145457
                }
            }
        }
    }
}
```

Example API call to get Malware scoring

```
curl  - X POST https: //caws-dev.nsslabs.com/graphql   -d  'query=query
{
        securityProfileAnalysis
               (
                      start:
                      "2013-08-02",
                      end:   "2017-08-31",
                      account:   4,
                      profile:   18
               ) {
                      id
                      name
                      pagedThreats
                             (
                                    excludeBlocked: true,
                                    excludeBypassed: false,
                                    sortBy:   testDate,
                                    sortDescending:   true
                             )  {
                                    pagingInfo {
                                           total
                                           pageCount
                                           currentPage
                                    }
                                    nodes {
                                           nssid
                                           testDate
                                           discoveryDate
                                           device {
                                                  id
                                                  name
                                                  type
                                                  vendor   {
                                                         id
                                                         name
                                                  }
                                           }
                                           exploitBlocked
                                           captureDroppedMalware
                                           malwareBlocked
```

```
                                        processCount
                                        maliciousProcessCount
                                        fileCount
                                        connectionCount
                                    }
                                }
                            }
}
'  -H '  authorization: bearer  < API Token > '
```

JSON Response

```
{
    "data": {
        "securityProfileAnalysis": {
            "id": "18",
                    "name": "Windows Defender-allapp-allplat",
                    "pagedThreats": {
                "pagingInfo": {
                    "total": 37,
                                    "pageCount": 1,
                                    "currentPage": 1
                },
                                  "nodes": [                  {
                                "nssid": "NSS-2017-3TXTG5",
                                                    "testDate": "2017-08-30T01:27:45.000Z",
                                                    "discoveryDate": "2017-08-30T00:11:27.000Z",
                                                    "device": {
                                    "id": "165",
                                                            "name": "Windows Defender",
                                                            "type": "Endpoint Protection",
                                                            "vendor": {
                                        "id": "18",
                                                                    "name": "Microsoft"
                                    }
                                },
                                                    "exploitBlocked": false,
                                                    "captureDroppedMalware": false,
                                                    "malwareBlocked": null,
                                                    "processCount": 5,
                                                    "maliciousProcessCount": 0,
                                                    "fileCount": 0,
                                                    "connectionCount": 0
                            },                      {
                                "nssid": "NSS-2017-3TPKM6",
                                                    "testDate": "2017-08-30T01:27:53.000Z",
                                                    "discoveryDate": "2017-08-29T02:05:45.000Z",
                                                    "device": {
                                    "id": "165",
                                                            "name": "Windows Defender",
                                                            "type": "Endpoint Protection",
                                                            "vendor": {
                                        "id": "18",
                                                                    "name": "Microsoft"
                                    }

                            },
```

Example Python API call to get Global Threat Analytics for given date range:

```
import json
import requests

# ...
#
# Function to Get Global Analytics (Capture Data).
def get_global_analytics(start, end, token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            query ThreatAnalytics($start: String!, $end: String!){
                threatAnalytics(start: $start, end: $end){
                    applicationThreatCounts{
                        application{
                            id
                            name
                            family
                            vendor{
                                id
                                name
                            }
                        }
                        count
                    }
                    countryThreatCounts{
                        countryCode
                        count
                    }
                    platformThreatCounts{
                        platform{
                            id
                            name
                            family
                            vendor{
                                id
                                name
                            }
```

```
                    }
                }
                metrics{
                    exploits{
                        count
                    }
                    scannedFiles{
                        count
                    }
                    scannedUrls{
                        count
                    }
                }
            }
        }
        """

# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'start': start,
    'end': end
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
# call to the "authenticateUser" Mutation. See this manual for an example
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
}

# Define Request Data to Send to Server.
#
# This is both the query and any variables in requires. Remember, GraphQL
# is JSON based so the variables parameter in the request has to be a JSON
# string!
data = {
    'query': query,
    'variables': json.dumps(variables)
}

# Send Request to Server.
#
# Unlike traditional REST APIs, in GraphQL, there is always a single
# endpoint you send the request to. The query you send as part of the
# request determines what the server actually does.
#
# The response body returned by the server is always formatted as JSON!
```

```
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('threatAnalytics')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

SEARCH FOR GLOBAL THREATS

This query allows you to search for global threats, not tied to a specific profile, pertaining to drive-by-exploits that have been detected by CAWS. There is currently no support to search for file exploits. Results returned by this query are batched in pages. Pagination information is returned in the result and client code is expected to examine that information to iterate through all the pages in the result. This query is equivalent to the /Captures/List endpoint in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
start	string	The start date you want to retrieve data from, in ISO-8601 format. Example: "2017-03-01"
end	string	The end date you want to retrieve data to, in ISO-8601 format. Example: "2017-04-01"
applications	int[], array of integers	An optional array of application ids you want to restrict data for. Example: [1,2,3]
platforms	int[], array of integers	An optional array of platform ids you want to restrict data for. Example: [1,2,3]
countries	string[], array of strings	An optional array of country codes, in ISO-3166 format, you want to restrict data for. Example: ["US"]
ipAddresses	string[], array of strings	an optional array of IP addresses you want to restrict data for. Example: ["66.127.0.1"]
processes	string[], array of strings	An optional array of MD5 hashes of files dropped by exploits and executed as processes you want to restrict data for. Example: ["ABC"]

Request and Response Structure

See the online documentation (https://caws3.nsslabs.com/docs/securityprofileoverview.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Example API call to get Search for Global Threat by date range

```
curl https://caws3.nsslabs.com/graphql -d 'query=query {
        threatInfo(start:  "2017-06-17", end:  "2017-06-17") {
        exploits
    {
            nssid
            discoveryDate
            processCount
            maliciousProcessCount
            connectionCount
        },
            connections
    {
            ipAddress,
```

```
            3.  port,
6.          },
            1.  processes
        {
        processName,
        md5Hash,
        sha1Hash,
        isMalicious
7.      }
8.  }  ' -H 'authorization: bearer <API Token>'
```

JSON Response

```
1.  {
2.      "data": {
3.          "threatInfo": {
4.              "exploits": [{
5.                  "nssid": "NSS-2017-38ZSFT",
6.                  "discoveryDate": "2017-06-17T00:05:10.000Z",
7.                  "processCount": 8,
8.                  "maliciousProcessCount": 0,
9.                  "connectionCount": 0
10.             }, {
11.                 "nssid": "NSS-2017-392YRV",
12.                 "discoveryDate": "2017-06-17T12:33:08.000Z",
13.                 "processCount": 8,
14.                 "maliciousProcessCount": 1,
15.                 "connectionCount": 1
16.             }, {
17.                 "ipAddress": "104.28.12.38",
18.                 "port": 80
19.             }, {
20.                 "ipAddress": "69.50.130.106",
21.                 "port": 80
22.             }],
23.             "processes": null
24.         }
25.     },
26.     "errors": [{
27.         "message": "Cannot read property 'length' of undefined",
28.         "status": "400",
29.         "code": "",
30.         "path": ["threatInfo", "processes"],
31.         "meta": {}
32.     }]
33. }
```

Example API Call to search for Global Threats by Date Range and IP Address.

```
1.  curl https://caws3.nsslabs.com/graphql -d 'query=query {
2.          threatInfo(start:  "2017-06-17", end:  "2017-06-
    17", ipAddresses: ["104.28.12.38"], platforms: [2, 3]) {
3.          exploits {
                1.  nssid
                2.  discoveryDate
                3.  processCount
                4.  maliciousProcessCount
                5.  connectionCount
4.          },      connections {
```

```
            ipAddress,       port,
        },     processes {
            processName,       md5Hash,       sha1Hash,       isMalicious
        }
    }
} ' -H 'authorization: bearer <API Token>' '
```

JSON Response

```
{
    "data": {
        "threatInfo": {
            "exploits": [{
                "nssid": "NSS-2017-38ZSFT",
                "discoveryDate": "2017-06-17T00:05:10.000Z",
                "processCount": 8,
                "maliciousProcessCount": 0,
                "connectionCount": 0
            }],
            "connections": [{
                "ipAddress": "104.28.12.38",
                "port": 80
            }],
            "processes": []
        }
    }
}
```

Example API Call to search for Global Threats by Date Range, IP Address and Platforms

```
curl https://caws3.nsslabs.com/graphql -d 'query=query {
        threatInfo(start:  "2017-06-1", end:  "2017-06-
17", ipAddresses: ["104.28.12.38"], platforms: [2, 3]) {
        exploits {
            nssid       discoveryDate       processCount       maliciousProcessCount       connectionC
ount
        },     connections {
            ipAddress,       port,
        },     processes {
            processName,       md5Hash,       sha1Hash,       isMalicious
        }
    }
}
' -H 'authorization: bearer <API Token>' '
```

JSON Response

```
{
    "data": {
        "threatInfo": {
            "exploits": [{
                "nssid": "NSS-2017-38XNJ3",
                "discoveryDate": "2017-06-16T18:20:49.000Z",
                "processCount": 8,
                "maliciousProcessCount": 0,
                "connectionCount": 0
            }, {
                "nssid": "NSS-2017-38ZSFT",
                "discoveryDate": "2017-06-17T00:05:10.000Z",
                "processCount": 8,
                "maliciousProcessCount": 0,
```

```
                "connectionCount": 0
            }],
            "connections": [{
                "ipAddress": "104.28.12.38",
                "port": 80
            }],
            "processes": []
        }
    }
}
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests

# ...
#
# Function to Search for Global Threats (Capture Data).
def find_global_threats(start, end, page, token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            query ThreatInfo($start: String!, $end: String!, $page: Int!){
                threatInfo(start: $start, end: $end){
                    pagedExploits(page: $page){
                                        nodes{
                                        nssid
                                        discoveryDate
                                        processCount
                                        maliciousProcessCount
                                        connectionCount
                                        }
                                        pagingInfo{
                                                currentPage
                                                total
                                                hasMore
                                        }
                                }
                                pagedConnections(page: $page){
```

```
                                        nodes{
                                        ipAddress
                                        port
                                        }
                                        pagingInfo{
                                                currentPage
                                                total
                                                hasMore
                                        }
                                }
                                pagedProcesses(page: $page){
                                        nodes{
                                        processName
                                        md5Hash
                                        sha1Hash
                                        isMalicious
                                        }
                                        pagingInfo{
                                                currentPage
                                                total
                                                hasMore
                                        }
                                }
                    }
                }
                """

# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'start': start,
    'end': end,
          'page': page
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
# call to the "authenticateUser" Mutation. See this manual for an example
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
}

# Define Request Data to Send to Server.
#
# This is both the query and any variables in requires. Remember, GraphQL
# is JSON based so the variables parameter in the request has to be a JSON
# string!
data = {
```

```
        'query': query,
        'variables': json.dumps(variables)
    }

    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('threatInfo')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

GET AN NSS ID'S DETAILS

This query allows you to retrieve information about a specific threat, identified by an NSS ID, that has been detected by CAWS. This query is equivalent to the /Captures/Get/{NSSId} endpoint in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
start	string	The start date you want to calculate metrics for the NSS ID relative to, in ISO-8601 format. This does not have to be the date the NSS ID was discovered. Example: "2017-03-01"
end	string	The end date you want to calculate metrics for the NSS ID relative to, in ISO-8601 format. This does not have to be the date the NSS ID was discovered. Example: "2017-04-01"
nssid	string	The NSS ID you want to retrieve data for. Example: "NSS-2017-38ZSFT"
reversingLabs	Json	Malware classification data.

Request and Response Structure

See the online documentation (https://caws3.nsslabs.com/docs/securityprofileoverview.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Example API Call to GET NSS ID Details

```
curl  https: //caws3.nsslabs.com/graphql -d 'query=query {
            threatIntel(start:   "2017-05-01",  end:   "2017-05-01",  nssid:   "NSS-2017-38ZSFT")  {
        nssid      details  {
            exploitType        discoveryDate        ipAddress        url        application  {
                application  {
                    id            name            family            vendor  {
                        id              name
                    }
                }
            }
            platform  {
                platform  {
                    id            name            family            vendor  {
                        id              name
                    }
                }
            }
```

```
            browser  {
                browser  {
                    id            name            family            vendor  {

                        id              name
                    }
                }
            }
        }
        shellcode  {
            disassembly
        }
        reversingLabs {
            id cloudAnalyticsThreatLevel md5Hash
    cloudAnalyticsThreatName
    sha256Hash coreAnalysis cloudAnalysisLastSeenDate cloudAnalysisFirstSeenDate
        }
    }
}
' -H '
authorization: bearer  < API Token > '
```

Response Structure

```
{
    "data": {
        "threatIntel": {
            "nssid":   "NSS-2017-38ZSFT",
                  "details": {
                "exploitType":   "url",
                        "discoveryDate":   "2017-06-17T00:05:10Z",
                        "ipAddress":   "104.28.12.38",
                        "url":   "http://ihackedit.com",
                        "application": {
                    "application": {
                        "id":   "176",
                                    "name":   "Adobe Flash Player 18.0.0.160",
                                    "family":   "Adobe Flash Player",
                                    "vendor": {
                                "id":   "1",
                                        "name":   "Adobe"
                        }
                    }
                },
                        "platform": {
                    "platform": {
                        "id":   "2",
                                    "name":   "Windows 7 SP1",
                                    "family":   "Windows",
                                    "vendor": {
                                "id":   "18",
                                        "name":   "Microsoft"
                        }
                    }
                },
                        "browser": {
                    "browser": {
                        "id":   "63",
                                    "name":   "Internet Explorer 9",
                                    "family":   "InternetExplorer",
                                    "vendor": {
                                "id":   "18",
```

```
                                        "name":   "Microsoft"
                        }
                    }
                }
            },
                    "shellcode": [        <SHELL WILL BE DISPLAYED HERE. Removed the shell code for better formatting of the document>     ]
        }
    }
}
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests

# ...
#
# Function to Get an NSS ID's Details (Capture Data)
def get_nssid(nssid, token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            query ThreatIntel($nssid: String!){
                threatIntel(nssid: $nssid){
                    nssid
                    details{
                        exploitType
                        discoveryDate
                        ipAddress
                        url
                        application{
                            application{
                                id
                                name
                                family
                                vendor{
                                    id
                                    name
                                }
                            }
```

```
                    }
                    platform{
                        platform{
                            id
                            name
                            family
                            vendor{
                                id
                                name
                            }
                        }
                    }
                    browser{
                        browser{
                            id
                            name
                            family
                            vendor{
                                id
                                name
                            }
                        }
                    }
                }
                shellcode{
                    disassembly
                }
                reversingLabs{
                    id
                    cloudAnalyticsThreatLevel
                    md5Hash
                    cloudAnalyticsThreatName
                    sha256Hash
                    coreAnalysis
                    cloudAnalysisLastSeenDate
                    cloudAnalysisFirstSeenDate
                }

            }
        }
        """

# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'nssid': nssid
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
```

```
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }

    # Define Request Data to Send to Server.
    #
    # This is both the query and any variables in requires. Remember, GraphQL
    # is JSON based so the variables parameter in the request has to be a JSON
    # string!
    data = {
        'query': query,
        'variables': json.dumps(variables)
    }

    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('threatIntel')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

Downloading HTTP Traffic, PCAP Files, Shell Code and Malware Payload Files

For any given NSS ID, you can download its associated HTTP Traffic, PCAP, and Shellcode files. Please note that not every NSS ID will have these files available.

Example API Calls to Download HTTP, PCAP and SAZ Files

Action	Method	Endpoint
Download HTTP Traffic File	GET	/graphql/downloads?type=SAZ&nssid={NSSId}
Download PCAP	GET	/graphql/downloads?type=PCAP&nssid={NSSId}
Download Shellcode	GET	/graphql/downloads?type=SHELL&nssid={NSSId}
Download Malware	GET	/graphql/downloads?type=PAYLOAD&hash={MD5Hash}

Request and Response Structure

See the online documentation(https://caws3.nsslabs.com/docs/threatintel.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Example Get Query to Download Shell Code

```
1. curl 'https://caws3.nsslabs.com/graphql/downloads?type=SHELL&nssid=NSS-2017-
   3H8M7D' - H  'authorization: bearer <API Token>'  - H  'content-type: application/json'
```

Example Query to Download SAZ File

```
1. curl  'https://caws3.nsslabs.com/graphql/downloads?type=SAZ&nssid=NSS-2017-
   3H8M7D' - H  'authorization: bearer <API Token>'  - H  'content-type: application/json'
```

Example Query to Download PCAP File

```
1. curl  'https://caws3.nsslabs.com/graphql/downloads?type=PCAP&nssid=NSS-2017-
   3H8M7D' - H  'authorization: bearer <API Token>'  - H  'content-type: application/json'
```

Example Query to Download Exploit Payload File

```
2. curl  'https://caws3.nsslabs.com/graphql/downloads?type=PAYLOAD&hash=<MD5 Hash of the file>
   - H  'authorization: bearer <API Token>'  - H  'content-type: application/json'
```

Here are examples using Python to download HTTP Traffic, PCAP and Shellcode, and malware payload files. The example defines functions that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests
```

```
# ...
#
# Function Download HTTP Traffic File.
def download_traffic_file(nss_id, token):
    # Define Download URL.
    #
    # Unlike other requests to the server, to download a file using the CAWS
    # API, you just need to send an HTTP request to the following endpoint and
    # indicate what kind of file you want.
    download_url =
'https://caws3.nsslabs.com/graphql/downloads?type=SAZ&nssid={}'.format(nss_id)

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }

    # Send Request to Server.
    #
    # The response body returned by the server is a binary stream! You can
    # choose to handle it however you want, like saving it to a local flat
    # file for example.
    response = requests.get(api_url, headers=headers)
    return response.content

# ...
#
# Function Download PCAP File.
def download_pcap_file(nss_id, token):
    # Define Download URL.
    #
    # Unlike other requests to the server, to download a file using the CAWS
    # API, you just need to send an HTTP request to the following endpoint and
    # indicate what kind of file you want.
    download_url =
'https://caws3.nsslabs.com/graphql/downloads?type=PCAP&nssid={}'.format(nss_id)

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }

    # Send Request to Server.
    #
    # The response body returned by the server is a binary stream! You can
```

```
    # choose to handle it however you want, like saving it to a local flat
    # file for example. For PCAPs, the binary stream is actually a ZIP stream,
    # because PCAPs can actually be quite big. You'll need to unzip the stream
    # locally.
    response = requests.get(api_url, headers=headers)
    return response.content

# ...
#
# Function Download Shellcode File.
def download_shellcode_file(nss_id, token):
    # Define Download URL.
    #
    # Unlike other requests to the server, to download a file using the CAWS
    # API, you just need to send an HTTP request to the following endpoint and
    # indicate what kind of file you want.
    download_url =
'https://caws3.nsslabs.com/graphql/downloads?type=SHELL&nssid={}'.format(nss_id)

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }

    # Send Request to Server.
    #
    # The response body returned by the server is a binary stream! You can
    # choose to handle it however you want, like saving it to a local flat
    # file for example.
    response = requests.get(api_url, headers=headers)
    return response.content

# ...
#
# Function Download Malware Payload Files
def download_malware_payload_file(md5_hash, token):
    # Define Download URL.
    #
    # Unlike other requests to the server, to download a file using the CAWS
    # API, you just need to send an HTTP request to the following endpoint and
    # indicate what kind of file you want.
    download_url =
'https://caws3.nsslabs.com/graphql/downloads?type=PAYLOAD&hash={}'.format(md5_ha
sh)

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
```

```
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
}

# Send Request to Server.
#
# The response body returned by the server is a binary stream! You can
# choose to handle it however you want, like saving it to a local flat
# file for example.
response = requests.get(api_url, headers=headers)
return response.content
```

GET PROFILE DETAILS

This query allows you to retrieve information about a specific profile. This includes all threats found to be targeting the applications in that profile, and all threats that are tested against the security products in that profile. This query is equivalent to the /Replays/List endpoint in the CAWS 2 API. Starting with CAWS 3, you can only retrieve information pertaining to security products relative to a specific profile associated with your account. This is because most metrics pertaining to security products are now computed relative to a specific profile. This allows you to retrieve information about a security product that is more relevant to your organization.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1
profile	int	The id of the profile you want to retrieve data for. You can only pass an id for a profile associated with your account. Passing any other value will result in an error. You can get a list of profile ids associated with your account by invoking the `securityProfiles`query. Example: 62
start	string	The start date you want to calculate metrics for the profile ID relative to, in ISO-8601 format. Example: "2017-03-01"
end	string	The end date you want to calculate metrics for the profile relative to, in ISO-8601 format. Example: "2017-04-01"

Request and Response Structure

See the online documentation (https://caws3.nsslabs.com/docs/securityprofilethreat.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Example API call to Get Profile Details

```
curl https://caws3.nsslabs.com/graphql --data-urlencode 'query=query {
        securityProfileAnalysis(start:  "2017-06-15", end:  "2017-06-
15", account: 1, profile: 62) {
        id     securityProducts {
            securityProductResults {
                device {
                    id          name          type          vendor {
                        id            name
                    }
                }
                blockedCount        bypassingCount        totalCount        blockRate
            }
        }
```

```
        applications {
            targetedApplications {
                application {
                    id           name           family           vendor {
                        id             name
                    }
                }
                bypassingCount         totalCount
            }
        }
    }
}
' -H 'authorization: bearer <API Token>'
```

JSON Response

```
{
    "data": {
        "securityProfileAnalysis": {
            "id":   "62",
                  "securityProducts": {
                "securityProductResults": [          {
                    "device": {
                        "id":   "165",
                                    "name":   "Windows Defender",
                                    "type":   "Endpoint Protection",
                                    "vendor": {
                            "id":   "18",
                                            "name":   "Microsoft"
                        }
                    },
                                "blockedCount": 21,
                                "bypassingCount": 30,
                                "totalCount": 51,
                                "blockRate": 41.176
                }        ]
            },
                  "applications": {
                "targetedApplications": [          {
                    "application": {
                        "id":   "159",
                                    "name":   "Internet Explorer 10",
                                    "family":   "Internet Explorer",
                                    "vendor": {
                            "id":   "18",
                                            "name":   "Microsoft"
                        }
                    },
                                "bypassingCount": 18,
                                "totalCount": 25
                },  {
                    "application": {
                        "id":   "53",
                                    "name":   "Internet Explorer 9",
                                    "family":   "Internet Explorer",
                                    "vendor": {
                            "id":   "18",
                                            "name":   "Microsoft"
                        }
                    },
                                "bypassingCount": 1,
                                "totalCount": 9
```

```
47.              }       ]
48.           }
49.        }
50.     }
51. }
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests

# ...
#
# Function to Get a Profile's Data (Replay Data).
def get_profile_data(start, end, account, profile, token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            query SecurityProfileAnalysis($start: String!, $end: String!,
$account: ID!, $profile: ID!){
                securityProfileAnalysis(start: $start, end: $end, account:
$account, profile: $profile){
                    id
                    securityProducts{
                        securityProductResults{
                            device{
                                id
                                name
                                type
                                vendor{
                                    id
                                    name
                                }
                            }
                            blockedCount
                            bypassingCount
                            totalCount
                            blockRate
                        }
                    }
                    applications{
```

```
                    targetedApplications{
                        application{
                            id
                            name
                            family
                            vendor{
                                id
                                name
                            }
                        }
                        bypassingCount
                        totalCount
                    }
                }
            }
        }
        """

# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'start': start,
    'end': end,
    'account': account,
    'profile': profile
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
# call to the "authenticateUser" Mutation. See this manual for an example
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
}

# Define Request Data to Send to Server.
#
# This is both the query and any variables in requires. Remember, GraphQL
# is JSON based so the variables parameter in the request has to be a JSON
# string!
data = {
    'query': query,
    'variables': json.dumps(variables)
}

# Send Request to Server.
#
# Unlike traditional REST APIs, in GraphQL, there is always a single
# endpoint you send the request to. The query you send as part of the
# request determines what the server actually does.
```

```
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('securityProfileAnalysis')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

GET List of Devices

This query allows you to get list of security devices and IDs associated with the device names. The device IDs need to be used in the API calls to create security profiles etc. The response has been truncated for the convenience of documentation. The actual response for this query contains the list of security devices for which the account has access to.

GET Query to fetch list of devices:

```
curl https://caws3.nsslabs.com/graphql --data-urlencode 'query=query {
     devices(account: 1) {
        id,   name
    }
}
' -H 'authorization: bearer <API Token>'
```

Query Responet that contains list of devices:

```
{
    "data": {
        "devices": [{
            "id": "11",
            "name": "Emsisoft Internet Security 9.0"
        }, {
            "id": "12",
            "name": "ESET Smart Security 8.0"
        }]
    }
}
```

CREATE A PROFILE

This query allows you to create a new security profile. You have to create a profile to understand the threats bypassing or blocked by a security product. Below is a curl example of the API Call.

Post Query to Create a Profile

```
curl -X POST https://caws3.nsslabs.com/graphql -H 'authorization: bearer <API Token>'
    -H 'cache-control: no-cache' -H 'content-type: application/x-www-form-urlencoded' -
d 'query=mutation {
createSecurityProfile(account: 8, name: "Profile-
1", description: "Profile 1 Description", applications: ["1", "2", "3"], device: "171", platforms: ["1
", "2", "3"], location: "Profile 1 Location") {
    account name description applications {
        id name family __typename
    }
    device {
        id name type __typename
    }
    platforms {
        id name family __typename
    }
    location __typename
}
}  '
```

Query Response for Create a Profile

```
{
    "data": {
        "createSecurityProfile": {
            "account": "8",
            "name": "Profile-1",
            "description": "Profile 1 Description",
            "applications": [{
                "id": "1",
                "name": "Adobe AIR 1.0.4990",
                "family": "Adobe AIR",
                "__typename": "Application"
            }, {
                "id": "2",
                "name": "Adobe AIR 13.0.0.83",
                "family": "Adobe AIR",
                "__typename": "Application"
            }, {
                "id": "3",
                "name": "Adobe AIR 2.0.2.12610",
                "family": "Adobe AIR",
                "__typename": "Application"
            }],
            "device": {
                "id": "171",
                "name": "Cylance PROTECT",
                "type": "Advanced Endpoint Protection",
                "__typename": "Device"
            },
            "platforms": [{
                "id": "1",
                "name": "Windows 7",
```

```
                "family": "Windows",
                "__typename": "Platform"
            }, {
                "id": "2",
                "name": "Windows 7 SP1",
                "family": "Windows",
                "__typename": "Platform"
            }, {
                "id": "3",
                "name": "Windows 8",
                "family": "Windows",
                "__typename": "Platform"
            }],
            "location": "Profile 1 Location",
            "__typename": "SecurityProfileSummary"
        }
    }
}
```

Get List of Supported Applications, Browsers and Platform Packages

This query allows you to retrieve a list of supported application, browser, and platform packages supported for user submissions. You will primarily need this query to get a list of ids to pass as arguments to the appropriate parameters when you are submitting either a file or URL for scanning by CAWS. This is a new query that has no equivalent in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1
infold	int	Pass the same value as the `account` parameter

Request and Response Structure

See the online documentation (Applications, Browsers, Platforms) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Examples

```
curl https://caws3.nsslabs.com/graphql --data-urlencode 'query=query {
        applications (accountId: 1, infoId: 1) {
        applications {
            id,       name,       supportedExtensions
        },     browsers {
            id,       name
        },     platforms {
            id,       name
        }
    }
}
' -H 'authorization: bearer <API Token>'
```

Response

```
{
    "data": {
        "submissions": {
            "applications": [        {
                "id": 1,
                            "name":   "Adobe Reader DC 2015.007.20033",
                            "supportedExtensions":   ".pdf"
            },          {
                "id": 2,
```

```
                            "name":   "Adobe Reader 9.4",
                            "supportedExtensions":   ".pdf"
            },          {
                "id": 3,
                            "name":   "Adobe Reader DC 2015.020.20039",
                            "supportedExtensions":   ".pdf"
            },          {
                "id": 4,
                            "name":   "Quicktime 7.79",
                            "supportedExtensions":   ".mp3, .mp4 , .m4a, .wav, .avi, .mov"
            },          {
                "id": 5,
                            "name":   "Itunes 12.5.1",
                            "supportedExtensions":   ".mp3, .mp4 , .m4a, .wav, .avi, .mov"
            },  {
                "id": 10,
                            "name":   "Windows Media Player 12.0.10011",
                            "supportedExtensions":   ".mp3, .mp4 , .m4a, .wav, .avi, .mov"
            }      ],
                  "browsers": [        {
                "id": 1,
                            "name":   "Firefox 50.0.1"
            },          {
                "id": 2,
                            "name":   "Firefox 50.0.2"
            },  {
                "id": 6,
                            "name":   "Internet Explorer 11"
            }      ],
                  "platforms": [        {
                "id": 1,
                            "name":   "Windows 7"
            },          {
                "id": 2,
                            "name":   "Windows 10"
            }      ]
        }
    }
}
```

Here is an example using Python. The example defines a function that you can reuse in a Python program if you wish. The code comments are pretty much self explanatory:

```
import json
import requests

# ...
#
# Function to Get a List of Supported Application, Browser, and Platform
# Packages for User Submissions.
def get_submissions(account):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
```

```
# fields that we want returned in the response.
#
# For an online reference to all the different GraphQL Queries and
# Mutations supported by CAWS, including their entire response data
# structures, the online documentation at https://caws3.nsslabs.com/docs
# is your best friend.
query = """
        query Submissions($accountId: ID!, $infoId: ID!){
            submissions(accountId: $accountId, infoId: $infoId){
                applications{
                    id
                    name
                    supportedExtensions
                }
                browsers{
                    id
                    name
                }
                platforms{
                    id
                    name
                }
            }
        }
        """

# Define Query Variables.
#
# Variables are a cool and useful feature of GraphQL that help avoid ugly
# string concatenation when constructing queries that are sent to the
# server. We recommend you use variables whenever possible.
variables = {
    'accountId': account,
    'infoId': account
}

# Define Headers to Send to Server.
#
# The most important header to send to the server is the "Authorization"
# header that includes your authentication token acquired from a previous
# call to the "authenticateUser" Mutation. See this manual for an example
# on how to perform authentication.
headers = {
    'Authorization': 'bearer {}'.format(token)
}

# Define Request Data to Send to Server.
#
# This is both the query and any variables in requires. Remember, GraphQL
# is JSON based so the variables parameter in the request has to be a JSON
# string!
data = {
    'query': query,
    'variables': json.dumps(variables)
}
```

```
    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        results = data.get('submissions')
        return results
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('Query Failed.')
```

Submit a URL for Scan

This mutation allows you to submit a URL to CAWS for scanning. When you submit a URL to CAWS for scanning, you will receive an email indicating that the scanning operation has been completed. If, however, you are submitting URLs from a script, you will need to poll the system to determine when the results are available. We recommend that you poll the system after 5 to 10 minutes, as that is how long the scanning operation usually takes, depending on load. This query is equivalent to the /Scan/Url endpoint in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1
url	string	The URL you wish to submit for scanning. Example: "http://www.google.com"
platforms	int[], array of integers	An array of platform package ids you want to scan on. You can get a list of platform package ids by invoking the `submissions` query. Example: [1,2,3]
browsers	int[], array of integers	An array of browser package ids you want to scan using. You can get a list of browser package ids by invoking the `submissions` query. Refer to Browser Package reference table to get the browser ID

Return Value

An id identifying the submission. You can use the id to poll the system and retrieve the results associated with the submission when they are available. To poll the system and retrieve results when they are available, use the submissions query.

```
curl https://caws3.nsslabs.com/graphql --data-urlencode 'query=mutation {
uploadUrl(accountId: 1, url: "http://www.google.com", platforms: [1], browsers: [5])
        {
            id
        }
}
' -H 'authorization: bearer <API Token>'
```

Response for URL Submissions Post Query

```
[{
    "data": {
        "uploadURL": ["AB671C35586C4CC29C00ED3D7F6231C5"]
    }
}]
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests
import requests_toolbelt

# ...
#
# Function to Submit a URL.
def submit_url(url, account_id, platforms, browsers, token):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            mutation UploadUrl($account_id: ID!, $url: String!, $platforms:
[Int]!, $browsers: [Int]!){
                uploadUrl(accountId: $account_id, url: $url, platforms:
$platforms, browsers: $browsers)
            }
            """

    # Define Query Variables.
    #
    # Variables are a cool and useful feature of GraphQL that help avoid ugly
    # string concatenation when constructing queries that are sent to the
    # server. We recommend you use variables whenever possible.
    variables = {
        'account_id': account_id,
        'url': url,
        'platforms': platforms,
        'browsers': browsers
    }

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }
```

```
    # Define Request Data to Send to Server.
    #
    # This is both the query and any variables in requires. Remember, GraphQL
    # is JSON based so the variables parameter in the request has to be a JSON
    # string!
    data = {
        'query': query,
        'variables': json.dumps(variables)
    }
    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql', data=data,
headers=headers)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
        # ...
        #
        # We specified this above!
        submission_tokens = data.get('uploadUrl')
        return submission_tokens
    else:
        # Get Errors from Results.
        #
        # A failed response from the server will have a top most array of
        # objects in the JSON with an "errors" key.
        errors = results.get('errors')
        for error in errors:
            errorMessage = error.get('message', 'No Message in Error.')
            print(errorMessage)

        exit('URL Submit Failed.')
```

SUBMIT A FILE FOR SCAN

Unlike other requests to the server, to download a file using the CAWS API, you just need to send an HTTP request to an endpoint. When you submit a file to CAWS for scanning, you will receive an email indicating that the scanning operation has been completed. If, however, you are submitting files from a script, you will need to poll the system to determine when the results are available. We recommend that you poll the system after 5 to 10 minutes, as that is how long the scanning operation usually takes, depending on load. This query is equivalent to the /Scan/File endpoint in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1
file	string	Name of the file. Example: @testfile.pdf
platforms	int[], array of integers	An array of platform package ids you want to scan on. You can get a list of platform package ids by invoking the submissions query. Example: [1,2,3]
ApplicationPackage	int[], array of integers	The application to scan the file with. Use one of the values listed in the example and make sure to encode the value in the request. Refer to Application Package reference table to get the application ID

Return Value

An id identifying the submission. You can use the id to poll the system and retrieve the results associated with the submission when they are available. To poll the system and retrieve results when they are available, use the submissions query.

```
curl -X POST https://caws3.nsslabs.com/graphql
     -H 'authorization: bearer <API Token>'
   -H '  cache - control: no - cache '
   -H '  content - type: multipart / form - data;
boundary = -- --WebKitFormBoundary7MA4YWxkTrZu0gW '
   -H '  postman - token: 5113 cac1 - 82 c8 - 0e35 - 9 f49 - 763e d5499501 '
   -F '  platforms = [1]  '
   -F '  applications = [2]  '
   -F accountId=1
   -F '  operationName = "UploadFile"  ' \
   -F '  file = @CAWS User Guide3.0 _DRAFT_V1_amit1.pdf '
```

Response for File Submissions Post Query

```
[{
    "data": {
        "uploadFile": ["AB671C35586C4CC29C00ED3D7F6231C5"]
    }
}]
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests
import requests_toolbelt

def submit_file(file_name, file_path, file_mime_type, account_id, platforms,
applications, token):
    m = requests_toolbelt.multipart.encoder.MultipartEncoder(
        [
            ('files', (file_name, open(file_path, 'rb'), file_mime_type)),
            ('accountId', json.dumps(account_id)),
            ('platforms', json.dumps(platforms)),
            ('applications', json.dumps(applications))
        ]
    )

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token),
        'Content-Type': m.content_type
    }

    # Send Request to Server.
    #
    # Unlike other requests to the server, to submit a file using the CAWS
    # API, you just need to send an HTTP request to the following endpoint.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=m)
    results = response.json()
    if isinstance(results, list) and len(results) == 1:
        results = results[0]
    else:
```

```
    exit('Submit File Failed: response=\"{}\"'.format(str(results)))

data = results.get('data')
if data:
    submission_tokens = data.get('uploadFile')
    return submission_tokens
else:
    errors = results.get('errors')
    for error in errors:
        print(error.get('message', 'No Message in Error'))

    exit('Submit File Failed.')
```

GET URL/FILE SCAN STATUS

This query allows you to retrieve results for submitted files and URLs. You'll need to use this query in two ways. First, to poll the system to determine whether a submission is completed or not. Second, to retrieve the results for a completed submission. This query is equivalent to the Scan/Status/Url/{Token} in the CAWS 2 API.

Parameters

Parameter Name	Parameter Type	Parameter Description
account	int	Your account's id. You can only pass your account's id. Passing any other value will result in an error. Example: 1
infold	Int	Pass the same value as the `account` parameter

Request and Response Structure

See the online documentation(https://caws3.nsslabs.com/docs/securityprofilethreat.doc.html) for the complete request and response JSON structure for this query. As is the case for all GraphQL queries, only the fields you select in request JSON structure will be returned in the response JSON structure.

Post Method to Get File Submissions Results

```
curl -X POST https://caws3.nsslabs.com/graphql
-H 'authorization: bearer <API Token>' -H 'cache-control: no-cache' -H 'content-
type: application/json' d
'[
        {
        "query":"query fileDetailsInfo {submissions(accountId: 1, requestId: "7F613AA2FC9F4778A8B349116
        7B86742") {fileDetailsInfo{ id state displayName result hasExploit user platformName platformId
         applicationName applicationId utcSubmissionDate utcResponseDate }}}", "variables": null, "oper
        ationName": "fileDetailsInfo"
    }
]  '
```

Response for File Scan Results

```
[{
    "data":  {
        "submissions":  {
            "fileDetailsInfo":  {
                "id":  61,
                                    "state":  3,
                                    "displayName":   "DOC_pdf_small_.pdf",
                                    "result":  null,
                                    "hasExploit":  null,
                                    "user":   "apatel@nsslabs.com",
                                    "platformName":   "Windows 7",
                                    "platformId":  1,
                                    "applicationName":   "Adobe Reader 9.4",
                                    "applicationId":  20,
                                    "utcSubmissionDate":   "Wed Jul 12 2017 16:36:25 GMT+0000 (UTC)",
```

```
                                "utcResponseDate":   "Wed Jul 12 2017 16:36:26 GMT+0000 (UTC)"

            }
        }
    }
}]
```

Here is an example using Python. The example defines a function that you can reuse in a Python program. The code comments are pretty much self explanatory:

```
import json
import requests

# ...
#
# Function to Get User Submission Status and Results.
def get_my_submissions(account):
    # Define Query to Send to Server.
    #
    # The query that is sent to the server is either a GraphQL Query or a
    # GraphQL Mutation. In this example because we want to get data, a we are
    # sending a GraphQL Query.
    #
    # Remember, GraphQL is graph based and we have to explicitly specify the
    # fields that we want returned in the response.
    #
    # For an online reference to all the different GraphQL Queries and
    # Mutations supported by CAWS, including their entire response data
    # structures, the online documentation at https://caws3.nsslabs.com/docs
    # is your best friend.
    query = """
            query MySubmissions($accountId: ID!, $infoId: ID!){
                submissions(accountId: $accountId, infoId: $infoId){
                    pending{
                        id
                        displayName
                        type
                        typeName
                        info
                        platformName
                    }
                    urlInfo{
                        id
                        displayName
                        result
                        hasExploit
                    }
                    fileInfo{
                        id
                        displayName
                        displayHash
                        result
                        hasExploit
                    }
```

```
            }
        }
        """

    # Define Query Variables.
    #
    # Variables are a cool and useful feature of GraphQL that help avoid ugly
    # string concatenation when constructing queries that are sent to the
    # server. We recommend you use variables whenever possible.
    variables = {
        'accountId': account,
        'infoId': account
    }

    # Define Headers to Send to Server.
    #
    # The most important header to send to the server is the "Authorization"
    # header that includes your authentication token acquired from a previous
    # call to the "authenticateUser" Mutation. See this manual for an example
    # on how to perform authentication.
    headers = {
        'Authorization': 'bearer {}'.format(token)
    }

    # Define Request Data to Send to Server.
    #
    # This is both the query and any variables in requires. Remember, GraphQL
    # is JSON based so the variables parameter in the request has to be a JSON
    # string!
    data = {
        'query': query,
        'variables': json.dumps(variables)
    }

    # Send Request to Server.
    #
    # Unlike traditional REST APIs, in GraphQL, there is always a single
    # endpoint you send the request to. The query you send as part of the
    # request determines what the server actually does.
    #
    # The response body returned by the server is always formatted as JSON!
    response = requests.post('https://caws3.nsslabs.com/graphql',
headers=headers, data=data)
    results = response.json()

    # Get Data from Results.
    #
    # A successful response from the server will have a top most object in the
    # JSON with a "data" key. "data" will then have an object with a key named
    # for the Query or Mutation you requested. The structure of that object
    # will be the return value of the Query or Mutation you requested.
    # Remember, the online documentation at https://caws3.nsslabs.com/docs is
    # your best friend!
    data = results.get('data')
    if data:
```

```
    # ...
    #
    # We specified this above!
    results = data.get('submissions')
    return results
else:
    # Get Errors from Results.
    #
    # A failed response from the server will have a top most array of
    # objects in the JSON with an "errors" key.
    errors = results.get('errors')
    for error in errors:
        errorMessage = error.get('message', 'No Message in Error.')
        print(errorMessage)

    exit('Query Failed.')
```

GET EXPLOITS/MALWARE BYPASSING A PROFILE

This API call is used to get the list of Exploits blocked or bypassed by security products. The number of exploits bypassing one or more of your security products.

API CALL CURL EXAMPLES FOR PAGINATION AND BYPASS SCORING

```
curl  - X  POST https: //caws3.nsslabs.com/graphql \
         -H  'authorization: bearer <API Token>'
  - H  'content-type: application/x-www-form-urlencoded'
  - d  'query=query {
            securityProfileAnalysis(
            start:  "2017-08-30",
            end:  "2017-08-30",
            account:  4,
            profile:  925)
            {
    id  name  pagedThreats(excludeBlocked:  true,  excludeBypassed:  false,  page:  1,  sortBy:  testDate,  sortDescending:  true)  {
         pagingInfo  {
             total  pageCount  currentPage
         }
         nodes  {
             nssid  testDate  discoveryDate  device  {
                 id  name  type  vendor  {
                     id  name
                 }
             }
             exploitBlocked  captureDroppedMalware  malwareBlocked  processCount  maliciousProcessCount  fileCount  connectionCount
         }
    }
}
}    '
```

JSON Response

```
{
    "data": {
        "securityProfileAnalysis": {
            "id": "925",
            "name": "Windows Defender-165",
            "pagedThreats": {
                "pagingInfo": {
                    "total": 37,
                    "pageCount": 1,
                    "currentPage": 1
                },
                "nodes": [{
                            "nssid": "NSS-2017-3TXTG5",
                            "testDate": "2017-08-30T01:27:45.000Z",
                            "discoveryDate": "2017-08-30T00:11:27.000Z",
                            "device": {
                                "id": "165",
                                "name": "Windows Defender",
                                "type": "Endpoint Protection",
                                "vendor": {
```

```
                                    "id": "18",
                                    "name": "Microsoft"
                                }
                            },
                            "exploitBlocked": false,
                            "captureDroppedMalware": false,
                            "malwareBlocked": null,
                            "processCount": 5,
                            "maliciousProcessCount": 0,
                            "fileCount": 0,
                            "connectionCount": 0
                        },
```

API Reference Tables for URL and File Scan

This section covers the tables that provide the IDs for Application Packages, Browser Packages and Platforms. In the API calls if Browser or platform or application is the variable the ID has to be passed.

Application Package Reference Table

This reference table lists the application package IDs you need to pass as a parameter when submitting files to CAWS for scanning. Each application package supports a specific extension and you are expected to pass an application package ID appropriate for the extension of the file you are submitting.

Id	Application	Supported Extensions
1	Adobe Reader DC 2015.007.20033	.pdf
2	Adobe Reader 9.4	.pdf
3	Adobe Reader DC 2015.020.20039	.pdf
4	Quicktime 7.79	.mp3, .mp4 , .m4a, .wav, .avi, .mov
5	Itunes 12.5.1	.mp3, .mp4 , .m4a, .wav, .avi, .mov
6	Itunes 12.5.4	.mp3, .mp4 , .m4a, .wav, .avi, .mov
7	Microsoft Office 2013	.doc, .docx, .xls, .xlsx, .csv, .ppt, .pptx
8	Microsoft Office 2016	.doc, .docx, .xls, .xlsx, .csv, .ppt, .pptx
9	Windows Media Player 12.0.7601	.mp3, .mp4 , .m4a, .wav, .avi, .mov
10	Windows Media Player 12.0.10011	.mp3, .mp4 , .m4a, .wav, .avi, .mov
11	VLC 2.2.3	.mp3, .mp4 , .m4a, .wav, .avi, .mov
12	VLC 2.2.4	.mp3, .mp4 , .m4a, .wav, .avi, .mov

Browser Package Reference Table

This reference table lists the browser package IDs you need to pass as a parameter when submitting URLs to CAWS for scanning.

Id	Browser
1	Firefox 50.0.1
2	Firefox 50.0.2
3	Google Chrome 53.0.2785.101

Id	Browser
4	Google Chrome 54.0.2840.59
5	Internet Explorer 9
6	Internet Explorer 11

Platform Package Reference Table

This reference table lists the platform package IDs you need to pass as a parameter when submitting files or URLs to CAWS for scanning.

Id	Platform
1	Windows 7 SP1
2	Windows 10