PUBLIC

High-Tech Bridge's
Mobile App Scanner API Documentation
Version v1.8
01st of October 2018

PUBLIC

General Overview

API Documentation and How-To

API Specifications

Field Name	Value
Protocol	HTTP/HTTPS
Request Type	GET/POST
URL	https://www.htbridge.com/mobile/api/

Example of Transactions using CURL

 # Downloading app from Google Play and starting test:

$ curl --data 'app_id=com.viber.voip' https://www.htbridge.com/mobile/api/download_apk

 # Uploading APK/IPA file and starting test:

$ curl -F malware_check=0 -F hide_in_statistics=0 -F file=@diva-beta.apk https://www.htbridge.com/mobile/api/upload

 # Get test results:

In the previous example, if app is found and test is started, we will get a test ID in the response. Once we have test ID, we can query API for test results. We can query either by full ID (id) or by short ID (short_id).

$ curl https://www.htbridge.com/mobile/api/test_info/id/[TEST_ID]

 # Delete test (possible only for manually uploaded APK/IPA files):

$ curl http://www.htbridge.com/mobile/api/delete/id/[TEST_ID]

 # Refresh test by redownloading (possible only for APKs downloaded from Google Play)

$ curl https://www.htbridge.com/mobile/api/refresh/id/[TEST_ID]

The output is detailed below:

• **unprocessed:** the application is in the process of uploading, contains details of the API level of the uploader, the amount of tests made, the amount of tests in the queue, and the total estimated time of completion.

• **processing:** the application has been uploaded and is now being processed. The output will be populated with information such as if the test can be deleted, if its refreshable, the status, position in the queue and SAST/DAST results.

• **uploading**: the application is in the process of being uploaded. Output will show confirmation of the successful upload of an application chunk.

• **refreshing**: an existing application is being re-tested. The output will comprise of the packages device type, the number of files, total size, presence of a malware check, test ID, etc.

• **deleting**: an existing application is being deleted. The output will state if the operation was successful or not.

• **highlights**: details the most important findings of the test, such as if DAST was performed, and the test ETA.

• **downloading APK from Play Store:** invoking the option to automatically download an APK from the Google Play Store.

Unprocessed

Corresponds to details for when the application is in the process of uploading, such as API details, test ETA and intermittent SAST/DAST data. The structure is as follows:

Syntax:

"status":

• Syntax: string

• Always present

• Description: the status of the test, "unprocessed", etc

Highlights

The highlights of the test, including messages indicating that DAST was performed. The structure is as follows:

```
[
  {
    "status": bool,
    "result": {
      "class": string
      "message": string
      "total": int
    }
  },

{
    "status": bool,
    "result": {
          "message": string,
      "class": string,
    }
  }
]
```

"result":

• Syntax: {}

• Always present

• Description: a list holding information regarding high-level data, such as if DAST was performed, completion ETA, etc.

"class":

• Syntax: string

• Always present

• Description: the classification of the message, e.g warning

"message":

• Syntax: string

• Always present

• Description: indicates backend testing information

"total":

• Syntax: int

• Always present

• Description: the total number of messages

"status":

• Syntax: bool

• Always present

• Description: a boolean value indicating the presence of the message

"result":

• Syntax: {}

• Always present

• Description: a list holding information regarding high-level data

"class":

• Syntax: string

• Always present

• Description: the classification of the message, e.g. warning

"message":

• Syntax: string

• Always present

• Description: indicates DAST test information

Processing

Corresponds to details for when the application has been uploaded, and is undergoing processing, such as if the test id deletable, if its refreshable, the status, position in the queue and SAST/DAST information. SAST and DAST findings will be populated intermittently as the test progresses. The structure is as follows:

"status":

• Syntax: { }

• Always present

• Description: mentions the current test status, and an ETA

"audit_verbose":

• Syntax: bool

• Always present

• Description: indicates the verbosity of the information returned

The following structure **"data"** holds application and test information

"data":

• Syntax: {}

• Always present

• Description: holds application and test information

It is comprised of the following structures:

"app_info" is a list that holds information on the uploading app, such as name, version, device type.

"app_id":

• Syntax: string

• Always present

• Description: the ID of the application

"app_version":

• Syntax: string

• Always present

• Description: the version of the application

"device_type":

• Syntax: string

• Always present

• Description: the OS type the app will run on

"test_icon":

• Syntax: bool

• Always present

• Description: indicates if a test icon is present

"test_id":

• Syntax: string

• Always present

• Description: the reference ID of the test

"test_start":

• Syntax: int

• Always present

• Description: when the test was started

"test_stop":

• Syntax: int

• Always present

• Description: when the test was stopped

"test_apis" is a structure that corresponds to results returned from other free service APIS. It contains arrays for radar, WEBSEC and SSL respectively. The structure is as follows:

```
test_apis: {
        radar: [
                {
                 id: int,
                 country:,
                cybersquatting: 1,
                 id: string,
                 lat: float,
                 lng: float,
                 orig_url: string,
                  phishing: int,
                  server_ip: string,
                  short_id: string,
                  ts: float,
                  typosquatting: int
                }
               ],
         ssl: [
              {
                id: int,
                grade: string,
                has_ssl_tls: bool,
```

```
        hipaa: bool,
        hostname: string,
        http_response: string,
        ip: string,
        nist: bool,
        pci_dss: bool,
        port: int,
        reverse_dns: string,
        score: int,
        server_signature: string,
        short_id: string,
        ts: int
       }
      ],
websec: [
          {
           grade: string,
           hostname: string,
           http_response: string,
           id: string,
           redirect_to: string,
           reverse_dns: string,
           server_ip: string,
           server_signature: string,
           short_id: string,
           tested_url: string,
           ts: int;
           }
          ]
```

The first structure is **"radar"**, which is as follows:

"id":

• Syntax: string

• Always present

• Description: the ID corresponding to the RADAR test

"country":

• Syntax: string

• Always present

• Description: the country of the detected domain

"cybersquatting":

• Syntax: int

• Always present

• Description: indicates if a potential cybersquatting domain was returned

"lat":

• Syntax: float

• Always present

• Description: the latitude of the domain

"lng":

• Syntax: float

• Always present

• Description: the longtitude of the domain

"orig_url":

• Syntax: string

• Always present

• Description: the original URL of the domain

"phishing":

• Syntax: int

• Always present

• Description: indicates if any potential phishing domains were returned

"server_ip":

• Syntax: string

• Always present

• Description: the IP address of the server

"short_id":

• Syntax: string

• Always present

• Description: the short ID of the test

"ts":

• Syntax: float

• Always present

• Description: the timestamp of the test

"typosquatting":

• Syntax: int

• Always present

• Description: indicates if any potential typosquatting domains were returned

The second structure is **"ssl"**, which is as follows:

"grade":

• Syntax: string

• Always present

• Description: the grade of the tested domain

"has_ssl_tls":

• Syntax: bool

• Always present

• Description: is the tested domain has SSL/TLS

"hostname":

• Syntax: string

• Always present

• Description: indicates the hostname of the domain

"http_response":

• Syntax: string

• Always present

• Description: the returned HTTP response

"id":

• Syntax: string

• Always present

• Description: the ID of the test

"ip":

• Syntax: string

• Always present

• Description: the IP address of the domain

"nist":

• Syntax: bool

• Always present

• Description: indicates if the domain is compliant with NIST

"pci_dss":

• Syntax: bool

• Always present

• Description: indicates if the domain is compliant with PCI DSS

"port":

• Syntax: int

• Always present

• Description: the port that the test was carried out through

"reverse_dns":

• Syntax: string

• Always present

• Description: the reverse DNS record of the domain

"score":

• Syntax: int

• Always present

• Description: the scoring indicator of the test

"ts":

• Syntax: int

• Always present

• Description: the timestamp of the test

The third structure is **"websec"**, which is as follows:

"grade":

• Syntax: string

• Always present

• Description: the grade of the test

"hostname":

• Syntax: string

• Always present

• Description: the hostname of the domain

"http_response":

• Syntax: string

• Always present

• Description: the http response returned from the server

"id":

• Syntax: string

• Always present

• Description: the id of the test

"redirect_to":

• Syntax: float

• Present if the redirects are followed

• Description: indicates where the server redirects to

"reverse_dns":

• Syntax: string

• Always present

• Description: the reverse DNS record of the domain

"server_ip":

• Syntax: int

• Always present

• Description: the IP address of the server

"server_signature":

• Syntax: string

• Always present

• Description: the signature of the server

"short_id":

• Syntax: string

• Always present

• Description: the short ID of the test

"tested_url":

• Syntax: string

• Always present

• Description: the tested URL

"ts":

• Syntax: int

• Always present

• Description: the timestamp of the test

"test_behaviour" is a list that holds information on the device type and android specific areas in which tests are carried out, such as storage, location and contacts.

"data":

• Syntax: []

• Always present

• Description: an array that's hold each of the above-mentioned areas, which are detailed below

"storage":

• Syntax: string

• Always present

• Description: information regarding the android storage settings

"location":

• Syntax: string

• Always present

• Description: information regarding the android location settings

"contacts":

• Syntax: string

• Always present

• Description: information regarding the android contacts settings

"device_type":

• Syntax: string

• Always present

• Description: the device type, eg android

"test_dast" is a list that holds information about the DAST results of the test, such as intel and vulnerabilities. The structure is as follows:

```
test_dast: {
        app_info: {
                version: string,
                name: string,
                package: string
                }
        intel: {
                mitm_hosts: {
                    information: {
                       CVSSv3_score: string,
                        cwe: string,
                        description: string,
                        insecure_code: string,
                        iw_portal_ftype: string,
                        iw_portal_warningtype: string,
                        level: string,
                        links: [],
                        name: string,
                        owasp: string,
                        secure_code: string,
                     intel: { },
        vulns: {
                cleartext_sqlite_database: {
                    information: {
                        CVSSv3_score: string,
                        cwe: string,
                        description: string,
                        insecure_code: string,
                        iw_portal_ftype: string,
                        iw_portal_warningtype: string,
                        level: string,
                        links: [],
                        name: string,
                        owasp: string,
                        secure_code: string,
                     proof: {},

                hardcoded_info: {
                    information: {
                        CVSSv3_score: string,
                        cwe: string,
                        description: string,
```

```
                        insecure_code: string,
                        iw_portal_ftype: string,
                        iw_portal_warningtype: string,
                        level: string,
                        links: [],
                        name: string,
                        owasp: string,
                        secure_code: string,
                    vulns: {},
                }
            }
        }
}
```

"app_info" is a list that holds information on the uploading app, such as name, version, device type.

"name":

• Syntax: string

• Always present

• Description: the name of the application

"package":

• Syntax: string

• Always present

• Description: the application package name

"version":

• Syntax: string

• Always present

• Description: the application version

The remaining element of **"test_dast"** is "intel". Each discovery is presented in a list with the elements of "information" and "intel".

Each discovered DAST intel issue will follow the below structure:

"mitm_hosts":

• Syntax: { }

• Always present

• Description: lists remote hosts accessible by the mobile application

As mentioned the following "information" and "intel" sections are part of the "mitm_hosts" structure

"information":

• Syntax: { }

• Always present

• Description: details information such as CVSSv3 score, an example of the insecure code, OWASP details and warning type, as shown below.

The following resides in the above "information" list:

"CVSSv3_score":

• Syntax: string

• Always present

• Description: the CVSSv3 score if applicable

"cwe":

• Syntax: string

• Always present

• Description: the CVE score if applicable

"description":

• Syntax: string

• Always present

• Description: a description of the DAST issue

"insecure_code":

• Syntax: string

• Always present

• Description: an example of the insecure code

"iw_portal_ftype":

• Syntax: string

• Always present

• Description: the ImmuniWeb portal ftype

"iw_portal_warningtype":

• Syntax: bool

• Always present

• Description: the ImmuniWeb portal warning type

"links":

• Syntax: [ ]

• Always present

• Description: relevant links

"name":

• Syntax: string

• Always present

• Description: in this case, a list of the remote hosts

"owasp":

• Syntax: string

• Always present

• Description: relevant OWASP information

"secure_code":

• Syntax: string

• Always present

• Description: an example of secure code

The remaining section in "mitm_hosts" is the "intel" list. It is an array holding the IP address and reverse DNS of the man-in the-middle host.

```
Intel: [
        ip_address: string,
        hostname: string,
        ]
```

The next section of "test_dast" is "vulns", which is a list containing details on the found DAST vulnerabilities within the application.

Each discovered DAST intel issue will follow the below structure:

"cleartext_sqlite_database":

• Syntax: { }

• Always present

• Description: contains the information related to the DAST vulnerability

As mentioned the following "information" and "proof" sections are part of the "cleartext_sqlite_database" structure:

"information":

• Syntax: { }

• Always present

• Description: details information such as CVSSv3 score, an example of the insecure code, OWASP details and warning type, as shown below.

The following resides in the above "information" list:

"CVSSv3_score":

• Syntax: string

• Always present

• Description: the CVSSv3 score if applicable

"cwe":

• Syntax: string

• Always present

• Description: the CVE score if applicable

"description":

• Syntax: string

• Always present

• Description: a description of the DAST issue

"insecure_code":

• Syntax: string

• Always present

• Description: an example of the insecure code

"iw_portal_ftype":

• Syntax: string

• Always present

• Description: the ImmuniWeb portal ftype

"iw_portal_warningtype":

• Syntax: bool

• Always present

• Description: the ImmuniWeb portal warning type

"links":

• Syntax: [ ]

• Always present

• Description: relevant links

"name":

• Syntax: string

• Always present

• Description: in this case, a list of the remote hosts

"owasp":

• Syntax: string

• Always present

• Description: relevant OWASP information

"secure_code":

• Syntax: string

• Always present

• Description: an example of secure code

The "proof" section is a list of evidence of the found vulnerability.

proof: {
....
....
}

The same structure remains for any other discovered DAST vulnerabilities:

"hardcoded_info":

• Syntax: { }

• Always present

• Description: contains the information related to the DAST vulnerability

As mentioned the following "information" and "proof" sections are part of the "hardcoded_info" structure:

"information":

• Syntax: { }

• Always present

• Description: details information such as CVSSv3 score, an example of the insecure code, OWASP details and warning type, as shown below.

The following resides in the above "information" list:

"CVSSv3_score":

• Syntax: string

• Always present

• Description: the CVSSv3 score if applicable

"cwe":

• Syntax: string

• Always present

• Description: the CVE score if applicable

"description":

• Syntax: string

• Always present

• Description: a description of the DAST issue

"insecure_code":

• Syntax: string

• Always present

• Description: an example of the insecure code

"iw_portal_ftype":

• Syntax: string

• Always present

• Description: the ImmuniWeb portal ftype

"iw_portal_warningtype":

• Syntax: bool

• Always present

• Description: the ImmuniWeb portal warning type

"links":

• Syntax: [ ]

• Always present

• Description: relevant links

"name":

• Syntax: string

• Always present

• Description: in this case, a list of the remote hosts

"owasp":

• Syntax: string

• Always present

• Description: relevant OWASP information

"secure_code":

• Syntax: string

• Always present

• Description: an example of secure code

The "vulns" section is a list of evidence of the found vulnerability.

```
vulns: {
        emails: [],
        passwords: [],
        tokens: [],
        urls: [],
        username: [],
        }
```

The next section, **"test_sast"** is a list that holds information about the SAST results of the test, such as intel and vulnerabilities. The structure is as follows:

```
test_sast: {
        app_info: {
            name: string,
            package: string,
            version: string,
           },
          behaviour_data: {
              data: [
                storage: string,
                location: string,
                contacts: string,
                ],
              device_type: string,
           },
           intel: {
               application: {
                },
               components: {
                  activities[{}].
                  information: {},
                  intel: {},
                  keys: {},
                  providers: {},
                  receivers: {},
                  services: {},
                },
              interact_with_trustmanager: {},
              interesting_files: {},
              list_ciphers: {},
```

```
            list_libraries: {},
            network_methods: {},
            permissions: {},
            rw_internal_storage: {},
            use_of_antiemulation: {},
            use_of_base64: {},
            use_of_command: {},
            use_of_keystore: {},
            use_of_socket: {},
            use_of_sql: {},
            use_of_webview: {},
            webview_can_access_providers: {},
            webview_file_uri_can_access_filesystem: {},
        },
        vulns: {
            create_temp_file: {},
            deserialize_object: {},
            execute_raw_sql_with_inputs: {},
            hardcoded_info: {},
            information_exposure: {},
            load_code_dynamically: {},
            missing_antiemulation: {},
            network_security_configuration_not_present: {},
            tapjacking_protection: {},
            use_of_backup: {},
            use_of_implicit_intent; {},
            use_of_intent-filter: {},
            use_of_unenrypted_network_protocols: {},
            use_of_weak_hasking_algorithm: {},
            use_of_weak_rng: {},
            vulnerable_activites: {},
            vunerable_receivers: {},
            vulnerable_services: {},
            webview_javascript_enabled: {},
            webview_use_of_setpluginstate: {},
        },
    },
```


The **"app_info"** section of **"test_sast"** that holds information on the uploading app, such as name, version and device type

"app_info":

- Syntax: {}

• Always present

• Description: holds the SAST test data as mentioned below

"name":

• Syntax: string

• Always present

• Description: the name of the application

"package":

• Syntax: string

• Always present

• Description: the application package name

"version":

• Syntax: string

• Always present

• Description: the application version

The next element of **"test_sast"** is **"behaviour_data",** which details the device type and data areas such as storage, location and contacts

"storage":

• Syntax: string

• Always present

• Description: information regarding the android storage settings

"location":

• Syntax: string

• Always present

• Description: information regarding the android location settings

"contacts":

• Syntax: string

• Always present

• Description: information regarding the android contacts settings

"device_type":

• Syntax: string

• Always present

• Description: the applications device type

The next element of **"test_sast"** is **"intel",** which details issues discovered in the app that are classified as 'intel'. The structure is as follows:

```
intel: {
        application: {},
        components: {},
        interact_with_trustmanager: {},
        interesting_files: {},
        list_ciphers: {},
        list_libraries: {},
        network_methods: {},
        permissions: {},
        rw_internal_storage: {},
        use_of_antiemulation: {},
        use_of_base64: {},
        use_of_command: {},
        use_of_keystore: {},
        use_of_socket: {},
```

```
        use_of_sql: {},
        use_of_webview: {},
        webview_can_access_providers: {},
        webview_domstorage_enabled: {},
        webview_file_uri_can_access_filesystem: {}
}
```

"application":

• Syntax: { }

• Always present

• Description: contains android specific application configuration details, such as allowBackup and vmSafeMode.

"components" is a list in the "intel" section of android components such as activities, services, receivers, keys, etc. The structure is as follows:

```
components: {
        activities: [{}],
        information: {},
        intel: {},
        keys: [{}],
        providers: [{}],
        receivers: [{}],
        services: [{}]
        }
```

"activities":

• Syntax: [{ }]

• Always present

• Description: a list of the observed android activities

"information":

• Syntax: { }

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links and OWASP details.

The "intel" part is the next with the parent "intel" section, which contains discovered intel issues in activities, providers, receivers and services. The structure is as follows:

```
intel: {
        activities: [{}],
        providers: [{}],
        receivers: [{}],
        services: [{}]
        }
```

"activities":

• Syntax: [{ }]

• Always present

• Description: a list of the observed android activities and details

"providers":

• Syntax: [{ }]

• Always present

• Description: a list of the observed android providers and details

"receivers":

• Syntax: [{ }]

• Always present

• Description: a list of the observed android receivers and details

"services":

• Syntax: [{ }]

• Always present

• Description: a list of the observed android services and details

The next part of “intel” is “keys”, which lists found keys regarding to activity, provider, receiver and service. The structure is as follows:

keys: [
activity,
provider,
receiver,
service
]

The next part of “intel” is “providers”, which details information regarding to found providers

"providers":

• Syntax: [{ }]

• Always present

• Description: a list of the observed providers and details

“Receivers” details information in regards to found receivers

"receivers":

• Syntax: [{ }]

• Always present

• Description: a list of the observed receivers and details

“services” details information in regards to found services

"services":

• Syntax: [{}]

• Always present

• Description: a list of the observed services and details

This concludes the “components” section of “intel”.

The next section of “intel” is “interact_with_trustmanger”, which lists details about TrustManger interactions. The structure is as follows:

```
interact_with_trustmanger: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found TrustManager instances

The next section of “intel” is “interesting_files”. The structure is as follows:

```
interesting_files: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found interesting files

The next section of “intel” is “list ciphers”. The structure is as follows:

```
list_ciphers: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found ciphers

The next section of “intel” is “list_libraries”. The structure is as follows:

```
list_libraries: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found libraries

"intel":

• Syntax: {}

• Always present

• Description: details of found ciphers

The next section of “intel” is “network_methods”. The structure is as follows:

```
network_methods: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found network methods, eg HttpUrlConnection, HttpsUrlConnection.

The next section of “intel” is “permissions”. The structure is as follows:

```
permissions:  {
        android: [{}],
        custom: [{}],
        groups: [],
        information: {},
        intel: {},
        keys: []
        }
```

"android":

• Syntax: [{}]

• Always present

• Description: details of android permissions

"custom":

• Syntax: [{}]

• Always present

• Description: details of custom permissions

"groups":

• Syntax: []

• Always present

• Description: details of found groups.

"information":

• Syntax: {}

• Always present

• Description: details of found groups.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances

"keys":

• Syntax: []

• Always present

• Description: details of found keys, eg permission, permission-group and users-permission.

The next section “rw_internal_storage”, lists details on the applications file system storage. The structure is as follows:

```
rw_internal_storage: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as getCacheDir() and getFileDir().

The next section “use_of_antiemulation”, lists details on the applications anti-emulation measures. The structure is as follows:

```
use_of_antiemulation: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found intel, such as isUserAMonkey().

The next section “use_of_base64”, lists details on the applications usage of base64. The structure is as follows:

```
use_of_base64: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as Base64.decode() and Base64.encodeToString().

The next section “use_of_command”, lists details on the applications usage of system commands. The structure is as follows:

```
use_of_command: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as getRuntime().exec().

The next section “use_of_keystore”, lists details on the applications usage of keystore. The structure is as follows:

```
use_of_keystore: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as KeyStore.getDefaultType().

The next section “use_of_socket”, lists details on the applications usage of sockets. The structure is as follows:

```
use_of_socket: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as Socket().

The next section “use_of_sql”, lists details on the applications usage of SQL. The structure is as follows:

```
use_of_sql: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as getWritableDatabase()

The next section “use_of_webview”, lists details on the applications usage of WebView. The structure is as follows:

```
use_of_webview: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as Webview.webview and new Webview()

The next section “webview_can_access_providers”, lists details on the applications usage of WebView and its access to providers. The structure is as follows:

```
webview_can_access_providers: {
        information: {},
        intel: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as setAllowContentAccess()

The next section “webview_dom_storage_enabled”, lists details on the applications usage of webview and its dom storage implementation. The structure is as follows:

webview_dom_storage_enabled: {
information: {},
intel: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as setDomStorageEnabled().

The next section “webview_file_uri_can_access_filesystem”, lists details on the applications usage of webview and its filesystem access implementation. The structure is as follows:

webview_file_uri_can_access_filesystem: {
information: {},
intel: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"intel":

• Syntax: {}

• Always present

• Description: details of found instances, such as setAllowFileAccess().

The following section of "test_sast" is "vulns", which is similar to "intel" but is a list of vulnerabilities discovered by the service. The structure is as follows:

```
vulns: {
        create_temp_file: {
                information: {
                   CVSSv3_score: string,
                   cwe: string,
                   description: string,
                   insecure_code: string,
                   iw_portal_ftype: string,
                   iw_portal_warningtype: string,
                   level: string,
                   links: [],
                   name: string,
                   owasp: string,
                   secure_code: string,
                   secure_code_OLD: string,
               proof: { }


                deserialize_object: { }
                   information:
                   CVSSv3_score: string,
                   cwe: string,
                   description: string,
                   insecure_code: string,
                   iw_portal_ftype: string,
                   iw_portal_warningtype: string,
                   level: string,
                   links: [],
                   name: string,
                   owasp: string,
                   secure_code: string,
                proof: { },


        execute_raw_sql_with_inputs: { }
                   information:
```

CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
proof: { },

load_code_dynamically{}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
proof: { },

missing_antiemulation: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
proof: [],

result: bool,
vulns: [],

network_security_configuration_not_present: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: [],
result: bool,

rw_external_storage: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

tapjacking_protection: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,

iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_backup: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_debug: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_hardcoded_credentials: {}
information:
CVSSv3_score: string,
cwe: string,

description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_implicit_intent: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_intent_filter: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_unencrypted_network_protocols: {}
information:

CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_weak_crypto_algorithm: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_weak_iv: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

use_of_weak_rng: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

vulnerable_providers: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

vulnerable_receivers: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: bool,
level: string,
links: [],
name: string,
owasp: string,

secure_code: string,
secure_code: string,
proof: {},

webview_javascript_cors_enabled: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

webview_javascript_enabled: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,
iw_portal_ftype: string,
iw_portal_warningtype: string,
level: string,
links: [],
name: string,
owasp: string,
secure_code: string,
secure_code: string,
proof: {},

webview_load_remote_url: {}
information:
CVSSv3_score: string,
cwe: string,
description: string,
insecure_code: string,

```
                    iw_portal_ftype: string,
                    iw_portal_warningtype: string,
                    level: string,
                    links: [],
                    name: string,
                    owasp: string,
                    secure_code: string,
                   secure_code: string,
                proof: {},
            }
        }
    }
}
```

"create_temp_file", lists details on the applications usage of creating temporary files. The structure is as follows:

```
create_temp_file: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as createTempFile()

"deserialize_object", lists details on the applications usage of object deserialization. The structure is as follows:

```
deserialize_object: {
        information: {},
```

proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as "implements Serializable"

“execute_raw_sql_with_inputs”, lists details on the applications usage of SQL with raw input. The structure is as follows:

execute_raw_sql_with_inputs: {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as "rawQuery()"

“hardcoded_info”, lists details on the applications usage of hardcoded data. The structure is as follows:

```
hardcoded_info: {
        information: {},
        vulns: {}                                                                    }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"vulns":

• Syntax: {}

• Always present

• Description: details of found instances, such as emails, passwords, usernames and network usage (http(s)).

The structure of the above "vulns" is as follows:

```
vulns: {
        emails: [],
        https://: {},
        passwords: {},
        usernames: {}
        }
```

“load_code_dynamically”, lists details on the applications usage of dynamically loading code. The structure is as follows:

load_code_dynamically: {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as "ClassLoader"

“missing_antiemulation”, lists details on the applications usage of missing antiemulation. The structure is as follows:

missing_antiemulation: {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"result":

• Syntax: bool

• Always present

• Description: indicates if the check returned a result

"proof":

• Syntax: {}

• Always present

• Description: details of found instances of the vulnerability

“network_security_configuration_not_present”, lists details on the applications usage of network security configuration. The structure is as follows:

```
network_security_configuration_not_present: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances of the vulnerability

"result":

• Syntax: bool

• Always present

• Description: indicates if the check returned a result

“rw_external_storage”, lists details on the applications usage of reading writing to external storage, such as SD cards. The structure is as follows:

```
rw_external_storage: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as getExternalFilesDir()

“tapjacking_protection”, lists details on the applications usage of TapJacking protection. The structure is as follows:

```
tapjacking_protection: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “android:filterTouchesWhenObscured=”true””

“use_of_backup”, lists details on the applications usage of backup. The structure is as follows:

```
Use_of_backup: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “android: allowBackup=”true””

“use_of_debug”, lists details on the applications usage of usage of debug. The structure is as follows:

```
use_of_debug: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “android:debuggable=”true””

“use_of_hardcoded_credentials”, lists details on the applications usage of usage of hardcoded credentials. The structure is as follows:

```
use_of_hardcoded_credentials: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “AES…”

“use_of_implicit_intent”, lists details on the applications usage of usage of implicit intent. The structure is as follows:

```
use_of_implicit_intent: {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “new Intent()”

“use_of_intent_filter”, lists details on the applications usage of usage intent filters. The structure is as follows:

```
use_of_intent_filter {
        information: {},
```

proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “<intent-filter>”

“use_of_unencrypted_network_protocols”, lists details on the applications usage of unencrypted network protocols. The structure is as follows:

use_of_unencrypted_network_protocols {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “HttpURLConnection”

“use_of_weak_crypto_algorithm”, lists details on the applications usage of weak cryptography. The structure is as follows:

```
use_of_weak_crypto_algorithm {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “AES/ECB/PKCS5Padding”

“use_of_weak_iv”, lists details on the applications usage of weak initialization vectors. The structure is as follows:

```
use_of_weak_iv {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “0, 0, 0, 0, 0, 0, 0, 0”

“use_of_weak_rng”, lists details on the applications usage of weak random number generators. The structure is as follows:

```
use_of_weak_rng {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “new Random()”

“vulnerable_providers”, lists details on the applications usage of vulnerable providers. The structure is as follows:

Vulnerable_providers {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances of the vulnerability.

“vulnerable_receivers”, lists details on the applications usage of vulnerable receivers. The structure is as follows:

vulnerable receivers {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances of the vulneraries

“webview_javascript_cors_enabled”, lists details on the applications usage of webview with JavaScript CORS enabled. The structure is as follows:

```
webview_javascript_cors_enabled {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “setAllowUniversalAccessFromFileURLs(true)”

“webview_javascript_enabled”, lists details on the applications usage of webview with JavaScript enabled. The structure is as follows:

```
Webview_javascript_enabled {
        information: {},
```

proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “setJavaScriptEnabled(true)”

“webview_load_remote_url”, lists details on the applications usage of webview loading a remote URL. The structure is as follows:

webview_load_remote_url {
information: {},
proof: {}
}

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “loadUrl(http)”

“webview_use_of_setpluginstate”, lists details on the applications usage of webviews with usage of setPluginState. The structure is as follows:

```
webview_use_of_setpluginstate {
        information: {},
        proof: {}
        }
```

"information":

• Syntax: {}

• Always present

• Description: details such as CVSSv3 scoring, warning type, reference links, OWASP details and an example of insecure code.

"proof":

• Syntax: {}

• Always present

• Description: details of found instances, such as “setPluginState()”

This concludes the “vulns” and the “test_sast” section.

This next section lists miscellaneous data about the test, such as queue position, and the test source.

"message":

• Syntax: string

• Always present

• Description: indicates the test status, eg if it has finished or not

"mitm":

• Syntax: string

• Always present

• Description: details mitm data

"status":

• Syntax: int

• Always present

• Description: indicates the test status, eg if it has finished or not

"test_apis":

• Syntax: bool

• Always present

• Description: indicates if APIs have been included in the test

"test_app_info":

• Syntax: bool

• Always present

• Description: indicates if application information is shown

"test_behaviour":

• Syntax: bool

• Always present

• Description: indicates if test behaviour is shown

"test_dast":

• Syntax: bool

• Always present

• Description: indicates if DAST has been performed

"test_debug":

• Syntax: bool

• Always present

• Description: indicates if test debug information is shown

"test_elapsed":

• Syntax: int

• Always present

• Description: indicates if he test has begun

"test_eta":

• Syntax: int

• Always present

• Description: indicates an ETA for the test

"test_queue_position":

• Syntax: int

• Always present

• Description: indicates the test queue position

"test_sast":

• Syntax: bool

• Always present

• Description: indicates if SAST was performed

"test_source":

• Syntax: bool

• Always present

• Description: indicates the source of the application

"total_in_queue_running":

• Syntax: int

• Always present

• Description: indicates the number of tests in the queue

"video":

• Syntax: bool

• Always present

• Description: indicates if the test utilized video

Uploading

The upload of a package to begin a test, an example and structure is as follows:

- Two additional fields can be checked upon upload:

malware_check:

0 - a virus-total malware check will not be performed

1 - a virus-check will be performed

hide_in_statistics:

0 - the test results will not be shown in the latest tests.

1 - the test results will be shown in the latest tests.

cURL Request:

```
curl "https://www.htbridge.com/mobile/api/upload?id=1528030162739&filename=1528030162739&chunk_number=1&chunk_size=1048576&total_size=9416950&total_chunks=8&hide_in_statistics=0&malware_check=1" -H "User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:61.0) Gecko/20100101 Firefox/61.0" -H "Accept: */*" -H "Accept-Language: en-GB,en;q=0.5" --compressed -H "Referer: https://www.htbridge.com/mobile/" -H "Content-Type: multipart/form-data; boundary=---------------------------11339781115505" -H "Cookie: __sharethis_cookie_test__=1; __auc=8501fe94165235b5942f335cde7; __unam=9ba74f1-165235b5a84-7ffda8c7-6; _ga=GA1.2.425771614.1533896514; tr_ref=https"%"3A"%"2F"%"2Fwww.google.com"%"2F; __asc=20a268251661c21efe0cdbda38e; _gid=GA1.2.2144381127.1538070278; __sharethis_cookie_test__=1" -H "Connection: keep-alive" --data ""
```

Response:

```
[
  {
     "status": "success",
     "message": "Successfully uploaded chunk."
   }
]
```

Refreshing

The re-testing of an existing test already present on the server. An example and structure is as follows:

cuRL Request:

```
curl "https://www.htbridge.com/mobile/api/refresh/id/p1bzAlgI?_=1538070626235" -H "User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:61.0) Gecko/20100101 Firefox/61.0" -H "Accept: */*" -H "Accept-Language: en-GB,en;q=0.5" --compressed -H "Referer: https://www.htbridge.com/mobile/" -H "X-Requested-With: XMLHttpRequest" -H "Cookie: __sharethis_cookie_test__=1; __auc=8501fe94165235b5942f335cde7; __unam=9ba74f1-
```

```
165235b5a84-7ffda8c7-6; _ga=GA1.2.425771614.1533896514;
tr_ref=https"%"3A"%"2F"%"2Fwww.google.com"%"2F; __asc=20a268251661c21efe0cdbda38e;
_gid=GA1.2.2144381127.1538070278; __sharethis_cookie_test__=1" -H "Connection: keep-alive"
```

Response:


```
[
 {
    "status": "success",
    "message": "Validation successful",
    "device_type": "android",
    "sha256": "7224fb114f6e75702eb946c194de170d5a32e4f3ab33dfe56ec0f44477069537",
    "total_files": 1562,
    "total_size": 33487740,
    "core_size": 17733516
  },
  {
    "status": "success",
    "id": "7224fb114f6e75702eb946c194de170d5a32e4f3ab33dfe56ec0f44477069537",
    "short_id": "kohS74Iy",
    "device_type": "android",
    "ts_start": 1538130953,
    "package_total_files": 1562,
    "package_total_size": 33487740,
    "package_core_size": 17733516,
    "show_test_results": 1,
    "malware_check": 0,
    "test_source": "internet"
  }
]
```

Deleting

The deletion of an existing test from the server. An example and the structure is as follows:

cURL Request:

```
curl 'https://www.htbridge.com/mobile/api/delete/id/lk5IAtAB?_=1538129072134' -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) gecko/20100101 Firefox/62.0' -H 'Accept: */*' -H 'Accept -Language: en-GB,en;q=0.5' --compressed -H 'Referer: https://htbridge.com/mobile/?id=lk5IAtAB' -H 'X-Requested-With: XMLHTTPRequest' -H 'DNT: 1' -H ' Connection: keep-alive'
```

Response:

```
{
  "status": true.
  "message": "Test has been deleted."
```

Download from Play Store

This section describes the process of using the "download from play store" feature present on the Hi-Tech Bridge Mobile App Scanner interface. The output consists of an array with boolean values indicating success of validation of the application package. Upon success, a list of details such as the device type, sha256

hash, total number of files, and file size. Upon unsuccessful validation, the details will show the id, the message and the status.

The structure is as follows:

```
[
        {
                "status": string,
                 "message": string,
                 "device_type": string,
                 "sha256":" string,
```

```
                "total_files": int,
                "total_size":int,
                "core_size": int
},
{
                "status": "error",
                "error_id": 0,
                "message": "This file was uploaded previously and is pending test.",
                "id": "cbb12519fce33cf5f2e17e92d919356038b3c2b4e49d11f61b51ad5b7cd1796e",
                "short_id": "e2rdRj6C",
                "dbg": [
                {
                        "id": string,
                        "short_id": string,
                        "test_source": string,
                        "score":" string,
                        "grade": string,
                        "app_name":string,
                        "app_id":string,
                        "app_version": string,
                        "device_type": string,
                        "test_sast": string,
                        "test_dast":" string,
                        "test_behaviour": string,
                                "test_apis": string,
                                "test_debug": string,
                                "package_total_files ": string,
                                "package_total_size": string,
                                "package_core_size": string,
                                "show_test_results": string,
                                "ts_start": string,
                                "ts_stop": string,
                                "user_ip": string,
                                "user_agent": string,
                                "user_city": string,
                                "user_country": string,
                                "verbose_audit": string,
                        }
                ]
        }
```

]

Below is the corresponding details in relation to successful validation:

"status":

• Syntax: string

• Always present

• Description: indicates if the validation was successful

"message":

• Syntax: string

• Always present

• Description: the presented message of a successful validation

"device_type":

• Syntax: string

• Always present

• Description: the applications device type, eg android

"sha256":

• Syntax: string

• Always present

• Description: the SHA256 hash of the application

"total_files":

• Syntax: int

• Always present

• Description: the total amount of files

"total_size":

• Syntax: int

• Always present

• Description: the total size of the application package

"core_size":

• Syntax: int

• Always present

• Description: the core size of the application package

Below is the corresponding details in relation to unsuccessful validation:

"status":

• Syntax: string

• Always present

• Description: indicates if the validation was successful

"error_id":

• Syntax: int

• Always present

• Description: the ID of the error

"message":

• Syntax: string

• Always present

• Description: the error message

"id":

• Syntax: string

• Always present

• Description: the ID of the test

"short_id":

• Syntax: string

• Always present

• Description: the short ID of the test

The following section is debug information for the unsuccessful test:

"id":

• Syntax: string

• Always present

• Description: the ID of the test

"short_id":

• Syntax: string

• Always present

• Description: the short ID of the test

"test_source":

• Syntax: string

• Always present

• Description: the source of the application, eg internet

"score":

• Syntax: string

• Always present

• Description: the scoring of the test

"grade":

• Syntax: string

• Always present

• Description: the grading of the test

"app_name":

• Syntax: string

• Always present

• Description: the name of the application

"app_id":

• Syntax: string

• Always present

• Description: the application ID

"app_version":

• Syntax: string

• Always present

• Description: the version of the application

"device_type":

• Syntax: string

• Always present

• Description: the applications device type, eg android

"test_sast":

• Syntax: string

• Always present

• Description: SAST test information for the application

"test_dast":

• Syntax: string

• Always present

• Description: DAST test information for the application

"test_behaviour":

• Syntax: string

• Always present

• Description: the behaviour information of the test

"test_apis":

• Syntax: string

• Always present

• Description: the test APIs

"test_debug":

• Syntax: string

• Always present

• Description: debug information regarding the test

"package_total_files":

• Syntax: string

• Always present

• Description: total number of files in the package

"package_total_size":

• Syntax: string

• Always present

• Description: the total size of the package

"package_core_size":

• Syntax: string

• Always present

• Description: the core size of the package

"show_test_results":

• Syntax: string

• Always present

• Description: if the test results will be shown

"ts_start":

• Syntax: string

• Always present

• Description: a timestamp of when the test was started

"ts_stop":

• Syntax: string

• Always present

• Description: a timestamp of when the test was stopped

"user_ip":

• Syntax: string

• Always present

• Description: the IP address of the uploader

"user_agent":

• Syntax: string

• Always present

• Description: the user agent of the uploader

"user_city":

• Syntax: string

• Always present

• Description: the city of the uploader

"user_country":

• Syntax: string

• Always present

• Description: the country of the uploader

"verbose_audit":

• Syntax: string

• Always present

• Description: the verbosity of the audit

Appendix 1: List of Messages and Error values

ID	Value
0	This file was uploaded previously and is pending test.

1	There was an error in uploading file.
2	The uploaded application is corrupted. Please double check file integrity.
4	The uploaded application is too large and cannot be processed.
5	The uploaded file is not a valid application. Please double check and try again.
6	Package not runnable.
7	No file provided for upload.
8	Error in upload.
9	Unable to create destination file.
10	Error saving file chunk.
20	Not logged in.
21	Unknown error, please try again.
22	Test can't be deleted.
25	Test can not be refreshed. Test is locked.
26	Test can not be refreshed because it's manually uploaded.
27	Test information can not be refreshed because the test is not finished yet.
30	Error in deleting, please try again later.
32	This E-Mail has been already added.
33	Error adding e-mail to this test.
34	Error sending e-mail.
35	E-mail has been sent.
36	All is ok!
37	E-mail added to this test.
38	All fields must be numeric.
40	Sorry, our systems are very busy now, we are working on the issue. Please try again in a few minutes.
41	You have tried to perform N tests in the last 3 minutes. The system is currently busy, please try again a bit later
42	You have tried to perform N tests in the last 24 hours. The system is currently busy, please try again a bit later.
43	Your IP is blacklisted.
44	Sorry, your API key is invalid or has expired. Please double-check it or contact us.
45	You reached the limit of N concurring running tests. Please wait until at least one of them is finished.
46	The application will be eligible for a re-scan after N hours