glorifiedgrep

Oct 27, 2020

Android Module

1 Android Module **1**
1.1 GlorifiedAndroid class . . . 1

2 CodeAnalysis class **63**

3 ParseManifest class **103**

4 OWASPAnalysis class **113**

5 OtherAnalysis class **121**

6 MalwareBehaviour class **127**

7 Utils class **165**
7.1 JKS class . . . 165
7.2 BKS class . . . 166
7.3 NativeELFAnalysis class . . . 167
7.4 NativeDEXAnalysis class . . . 168
7.5 SQL class . . . 170
7.6 Utils class . . . 171

8 GreppedOut class **173**

9 Resources **175**

10 Indices and tables **177**

Index **179**

CHAPTER 1

Android Module

This section shows the docs for all the methods that are available inthe GlorifiedAndroid class. This class extends various other others. Refer to those for complete documentation.

1.1 GlorifiedAndroid class

class glorifiedgrep.**GlorifiedAndroid**(*apk_path: str = None*, *output_dir: str = None*, *project_dir: str = None*, *rg_path: str = 'rg'*, *jadx_path: str = 'jadx'*, *clean_dir: bool = False*)

Main class that is instantiated when using GlorifiedAndroid.

__init__(*apk_path: str = None*, *output_dir: str = None*, *project_dir: str = None*, *rg_path: str = 'rg'*, *jadx_path: str = 'jadx'*, *clean_dir: bool = False*)

The init method for the whole GlorifiedAndroid module. This is interted throughout

Parameters

- **apk_path** (*str*) – Path to the APK
- **output_dir** (*str*) – Output dir for decompilation and unzipping, defaults to /tmp/glorified_android
- **project_dir** (*str*) – Project directory used for already decompiled and processed apks, defaults to None
- **rg_path** (*str*) – path to ripgrep. Defaults to looking for it in path
- **jadx_path** (*str*) – path to jadx. Defaults to looking for it in path
- **clean_dir** (*bool*) – delete the output directory before processing

Raises

- **NotValidPythonVersion** – Raises if python version 3 is not used
- **DifferentAPKExists** – Raises if decompiled APK is different than what is already decompiled

- **DependentBinaryMissing** – Raises if ripgrep, or jadx is not found

```
>>> # The default output directory is temp/GlorifiedAndroid folder. This can
↪be
>>> # overriden using output_dir='some/path'
>>> a = GlorifiedAndroid('/path/to/apk', output_dir='/out/dir')
```

Typically, the prefix for the file path is removed when processing filepaths in the various code analysis classes. This can be adjusted using

```
>>> a.remove_dir_prefix = ''
```

If **ripgrep** or **jadx** is not in path, analysis will not be complete. To pass a user defined path for either jadx or rg, the GlorifiedAndroid class can be instantiated as follows.

```
>>> a = GlorifiedAndroid('/path/to/apk', jadx_path='path/to/jadx', rg_path='/
↪path/to/rg')
```

all_cert_analysis()

Property runs all available checks in _CertAnalysis

Returns Dictionary of all cert analysis

Return type dict

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_manifest_analysis()
```

all_file_analysis()

Property runs all available checks in _FileAnalysis

Returns Dictionary of all analysis

Return type dict

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_file_analysis()
```

all_manifest_analysis() → dict

Property runs all available checks in _ManifestAnalysis

Returns Dictionary of all analysis

Return type dict

all_other_analysis()

Property runs all available checks in _OtherAnalysis

Returns Dictionary of all other analysis

Return type dict

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_other_analysis()
```

all_owasp_analysis()

Property runs all available checks in _OwaspMasvs

Returns Dictionary of all other analysis

Return type dict

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_owasp_analysis()
```

cert_bits() → int

Certificate bit

Returns Certificate bits

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_bits()
```

cert_certificate() → glorifiedgrep.out.GreppedOut

Returns a PEM encoded certificate

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_certificate()
```

cert_digest() → dict

Returns the digest hash in md5. sha1 and sha256

Returns Dictionary of hashes

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_digest()
```

cert_issuer() → glorifiedgrep.out.GreppedOut

The entity that verified the information and signed the certificate

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_issuer()
```

cert_public_key() → glorifiedgrep.out.GreppedOut

Get the public key from CERT.RSA

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_public_key()
```

cert_serial_number() → int

Used to uniquely identify the certificate within a CA's systems. In particular this is used to track revocation information

Returns Certificate serial number

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_serial_number()
```

cert_signature_algorithm() → str

The algorithm used to sign the public key certificate

Returns Algorithm used to create the certificate

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_signature_algorithm()
```

cert_subject() → list

The entity a certificate belongs to: a machine, an individual, or an organization.

Returns Dict of certificate subjects CN, O, C, ST, L, OU, Cn

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_subject()
```

cert_valid_dates() → dict

The that the certificate is valid before, after and if expired

Returns Dict of dates and if exipred

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_valid_dates()
```

cert_version() → int

The certificate version number

Returns Version number of the certificate

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_version()
```

code_accessibility_service(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses AccessibilityService and its various classes. It also looks for the accessibilityEvent method. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_accessibility_service()
```

code_add_javascriptinterface(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Leads to vulnerabilities in android version jellybean and below | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_add_javascriptinterface()
```

code_android_contacts_content_provider(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Idicates imports, or any other place where the ContactsContract class and its providors are being used. This typically indicates that the app can read various contact information from the phones contact list. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_android_contacts_content_provider()
```

code_apache_http_get_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpGet method from the apache library. This is generally used to make GET requests. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_get_request()
```

code_apache_http_other_request_methods(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpPut, HttpDelete, HttpHead, HttpTrace and HttpOptions methods from the apache library. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_apache_http_post_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpPost method from the apache library. This is generally used to make GET requests. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_api_builder(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method makes a best effort to detect api string builders within the decompiled Java code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_api_builder()
```

code_apk_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if calls to apk files are hardcoded.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apk_files()
```

code_aws_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where AWS queries are being made. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_aws_query()
```

code_base64_decode(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify base64 decode operations.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_base64_decode()
```

code_base64_encode(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify base64 encode operations.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_base64_encode()
```

code_boot_completed_persistance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses BOOT_COMPLETED action which is typically used to start a service or a receiver on reboot. This indicates persistance. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_boot_completed_persistance()
```

code_broadcast_messages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what broadcast messages are being sent in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_broadcast_messages()
```

code_broadcast_send(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify code that indicates broadcast messages being sent.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_broadcast_send()
```

code_browser_db_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the browser db. This db usually includes browsing history. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_browser_db_access()
```

code_byte_constants(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will create a dictionary of hardcoded byte constants.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_byte_constants()
```

code_call_log(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that retreives call logs. May be possible malware behaviour. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_call_log()
```

code_camera_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the camera and picture taking functionality. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_camera_access()
```

code_cipher_instance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of Cipher.getInstance in the decompiled source. class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_cipher_instance()
```

code_class_extends(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any classes that are extending another class.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_class_extends()
```

code_class_init(*class_name: str*, *show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will first identify import statemants from the provided `class_name` and then look for all new instances of `new class_name`. `class_name` can either be a class like Date, or a package name like java.utils.Date

Parameters

- **class_name** (*str*) – A valid class name. Can be either name; i.e. *Date*, or package name i.e *java.utils.Date*.
- **show_code** (*bool, optional*) – Show the full matched line, by default False, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_class_init()
```

code_clipboard_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where valies are being set or read from the clipboard. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_clipboard_manager()
```

code_command_exec(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all commands executed in shell using /bin/sh or .exec() in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_command_exec()
```

code_cookies (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where cookies are being set. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_cookies()
```

code_create_new_file (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that creates new files in the android system. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_new_file()
```

code_create_sockets (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

An InetSocketAddress is a special SocketAddress designed to represent the standard TCP Protocol address, so it thus has methods to set/query the host name, IP address, and Socket of the remote side of the connection (or, in fact the local side too) | Reference Android SDK | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_sockets()
```

code_create_tempfile (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all code which is using Java createTempFile | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_tempfile()
```

code_database_interaction(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is reading database files. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_database_interaction()
```

code_database_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that queries any database on the device. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_database_query()
```

code_debuggable_check(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for code what will check if the app is debuggable at run time. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_debuggable_check()
```

code_debugger_check(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for usage of isDebuggerConnected in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_debugger_check()
```

code_deserialization (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

ObjectInputSteam when used with 'readObject' 'readObjectNodData' 'readResolve' 'readExternal' will likely result in a Deserialization vulnerability | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_deserialization()
```

code_device_id (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where device id is being obtained. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_device_id()
```

code_device_serial_number (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for Build.SERIAL which can sometimes be used in addition with other things to build unique tokens. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_device_serial_number()
```

code_download_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses the DownloadManager class to download files from onlines services. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_download_manager()
```

code_dynamic_dexclassloader(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of DexClassLoader in the decompiled source. This can be used to execute code not installed as part of an application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_dynamic_dexclassloader()
```

code_dynamic_other_classloader(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of BaseDexClassLoader, SecureClassLoader, DelegateLastClassLoader, DexClassLoader, InMemoryDexClassLoader, PathClassLoader, URLClassLoader, Classloader in the decompiled source. This can be used to execute code not installed as part of an application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_dynamic_other_classloader()
```

code_exif_data(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects if the ExifInterface class is imported and then instantiated. This class is typically used to either set or get meta data from images | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_exif_data()
```

code_external_file_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where external files are being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_external_file_access()
```

code_file_observer(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of the FileObserver class being used. This class is used to check for file access or change and fire and event. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_file_observer()
```

code_file_read(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for FileInputStream within the decompiled Java code which would indicate which files the app is reading. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_file_read()
```

code_file_write(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for getByes() method which can indicate files being written by the app. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_write_file()
```

code_find_intents(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify intent builders.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_find_intents()
```

code_firebase_imports(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_firebase_imports()
```

code_get_environment_var(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for usage of getenv in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_get_environment_var()
```

code_google_api_keys(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Searches for Firebase or Google services API keys. It is likely that an app that uses Firebase will have keys in their sources, but these keys should be checked for what kind of access they allow.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_google_api_keys()
```

code_gps_location(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where GPS locations are being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_gps_location()
```

code_hashing_algorithms(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify hashing algorithms being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_hashing_algorithms()
```

code_hashing_custom(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify custom hashing algorithms being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_hashing_custom()
```

code_http_request_methods(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what HTTP request methods are being used. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_http_request_methods()
```

code_imports(*class_name: str*) → list

Returns an array of filepaths where a import statement matched the class_name. It does use a word boundary to get more of an exact match

Parameters **class_name** (*str*) – Name of the absolute or relative class

Returns List of file paths where a match has been found

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_imports()
```

code_intent_filters(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This identifies all the different types of intent filters

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_intent_filters()
```

code_intent_parameters(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify usage of the getStringExtra which is used to create parameters for intents. | Reference Android SDK | Reference OWASP

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_intent_parameters()
```

code_invisible_elements(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code will set the visibility of an element to invisible. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_invisible_elements()
```

code_jar_urlconnection(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is using the JarURLConnection API. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_jar_urlconnection()
```

code_js_read_file(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Gets or Sets whether JavaScript running in the context of a file scheme URL can access content from other file scheme URLs. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_js_read_file()
```

code_key_generator(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of KeyGenerator and its methods in the decompiled source. This class provides the functionality of a secret (symmetric) key generator | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_key_generator()
```

code_keystore_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where Bouncy castle bks or jks files are being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_keystore_files()
```

code_load_native_library(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method identifies where native libaries and loaded in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_load_native_library()
```

code_location(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
Identifies code that receives location information. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_location()
```

code_location_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
Identifies code that receives updated location information. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_location_manager()
```

code_logging(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method looks for the usage of Log class from Android SDK. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_logging()
```

code_make_http_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify when a HTTP connection is being made in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_make_http_request()
```

code_make_https_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify when a HTTPS connection is being made in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_make_http_request()
```

code_mediastore(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_mediastore()
```

code_notification_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that can access notifications. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_notification_access()
```

code_notification_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that controls notifications. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_notification_manager()
```

code_null_cipher(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify nullciphers are being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_null_cipher()
```

code_object_deserialization(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where cookies are being set. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_object_deserialization()
```

code_package_installed(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the usage of the getInstalledPackages method from the PackageManager class. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_parse_uri (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is parsing a URI. This could be related to web urls, or content provider urls. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_parse_uri()
```

code_password_finder (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify possible passwords in the code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_password_finder()
```

code_phone_sensors (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that initiates various sensors available by Android. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_phone_sensors()
```

code_rabbit_amqp (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Checks if Rabbit amqp imports are present

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_rabbit_amqp()
```

code_read_sms_messages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Searches for SmsMessage class which is typically used to read SMS messages send to a device. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_read_sms_messages()
```

code_reflection(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that allows reflections in Java. This is a finding. Refer to the references for the risk and usage of reflections. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_reflection()
```

code_regex_matcher(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is processing regex. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_regex_matcher()
```

code_regex_pattern(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that compiles regex patterns. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_regex_pattern()
```

code_root_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that indicates if the app requests su access.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_root_access()
```

code_screenshots(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies usage of Bitmap and BitmapFactory classes. Although these are for bitmap compression and manipulation, they are often used to take screenshots. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_screenshots()
```

code_sdcard(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify strings matching sdcard usage.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sdcard()
```

code_search(*regex: str, rg_options: str = '', show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Run any checks against the decompiled code. The regex should be in raw string format

Parameters

- **regex** (*str*) – Regex pattern
- **rg_options** (*str*) – ripgrep options, space seperated string, defaults to ''
- **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

code_send_sms_text(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code can send SMS/Text messages. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_send_sms_text()
```

code_services(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what services are being started or being bound to. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_services()
```

code_shared_preferences(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method discovers SharePreference and getSharePreference from the decompiled code. Interface for accessing and modifying preference data returned by Context.getSharedPreferences within the decompiled Java code. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_shared_preferences()
```

code_sim_information(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where device sim card information is being obtained. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sim_information()
```

code_sql_injection_points(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for execquery. If user input is used in this query, this will lead to SQL injection. | Reference | Reference | Reference | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_injection_points()
```

code_sql_injection_user_input(*show_code=False*)

Find places in code where a variable is being concatanated with a SQL statement

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns

- *GreppedOut* – GreppedOut object
- *Examples*
- ——— (*rtype: dict*)
- *>>> from glorifiedgrep import GlorifiedAndroid*
- *>>> a = GlorifiedAndroid('/path/to/apk')*
- *>>> a.code_sql_inection_points()*

code_sql_java_implementation(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any other SQL queries that are implemented in Java. This searches for .query, .insert, .update and .delete methods. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_java_implementation()
```

code_sql_query_other (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any other SQL queries like INSERT, DROP etc in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_query_other()
```

code_sql_select_raw_query (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any SELECT queries in the decompiled code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_select_raw_query()
```

code_sqlcipher_password (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This getWritableDatabase and the getReadableDatabase methods from sqlcipher classes (3rd party) takes the db password as their argument. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sqlcipher_password()
```

code_sqlite_operations(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This getWritableDatabase and the getReadableDatabase methods db instances for sqlite opertations. These calls can be followed to check what data is being entered in the database. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sqlite_operations()
```

code_ssl_connections(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if SSL is being used by the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_ssl_connections()
```

code_stack_trace(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where AWS queries are being made. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_stack_trace()
```

code_static_iv(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify static IV's. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_static_iv()
```

code_string_constants(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will create a dictionary of hardcoded string constants.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_string_constants()
```

code_stub_packed(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for indication that the application is packed.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_stub_packed()
```

code_system_file_exists(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects if the exists method from the File class is being called. This method is typically used to check if the path in the class constructor exists in the system. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_system_service(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify systemservices being called. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_system_service()
```

code_tcp_sockets(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify TCP sockets being opened by the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_tcp_sockets()
```

code_trust_all_ssl(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that willl allow all SSL connections to succeed without verifying the hostname. This is a finding. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_trust_all_ssl()
```

code_udp_sockets(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify UDP sockets being opened by the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_udp_sockets()
```

code_weak_hashing(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where weak hashing algorithems such as MD5, MD4, SHA1 or any RC hashes are used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_weak_hashing()
```

code_websocket_usage(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects common Websockets init classes. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_websocket_usage()
```

code_webview_content_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where the webview has can access data from a content provider. | Reference Android SDK | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_content_access()
```

code_webview_database(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This allows developers to determine whether any WebView used in the application has stored any of the following types of browsing data and to clear any such stored data for all WebViews in the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_database()
```

code_webview_debug_enabled(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks to see if debug is enabled in webview. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_debug_enabled()
```

code_webview_file_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where the webview has file access. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_file_access()
```

code_webview_get_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify webview get requests. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_get_request()
```

code_webview_js_enabled(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where JavaScript is enabled. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_js_enabled()
```

code_webview_post_request (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify webview get requests. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_post_request()
```

code_xml_processor (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify possible weaknesses in XML parsing and creation. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xml_processor()
```

code_xor_encryption (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for XOR encryption operation within the decompiled code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xor_encryption()
```

code_xpath (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if SSL is being used by the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xpath()
```

classmethod exodus_trackers (*trackers*)

Use this method to override the build in _TRACKERS constant with the response body from the exodus api. This is not recommended because some of the detection regex's from exodus are not valid. Example 'CrowdTangle': '.' The Exodus api url is https://reports.exodus-privacy.eu.org/api/trackers

Parameters **trackers** (*str*) – the json response body from the exodus api.

Examples

```
>>> import requests
>>> from glorifiedgrep.android.modules.constants import _Trackers
>>> res = requests.get('https://reports.exodus-privacy.eu.org/api/trackers').
↪text
>>> _Trackers().exodus_trackers(res)
```

file_activities_handling_passwords () → list

This method enumerates the xml files found in sources/res/layout/ and looks for the textPassword value to see which activities handle passwords.

Returns

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_activities_handling_passwords()
```

file_database_file_paths () → list

This method enumerates for sqlite database files, and returns a list of their paths

Returns a list of database file paths

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_database_file_paths()
```

file_get_file_types (*describe: bool = False, exclude: list = ["]*) → dict

Returns the magic values of all files found after unzipping the APK. Keys are sorted by mime values of the files

Parameters

- **describe** (*bool, optional*) – Get full description of file. Defaults to False
- **exclude** (*list, optional*) – Exclude the file extensions in an array. Defaults to None

Returns Dictionary of all files and their magic headers

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/file')
>>> a.file_get_file_types(exclude=['xml', 'png'])
```

file_get_java_classes() → list

Returns a list of found JAVA classes

Returns JAVA classes

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_get_java_classes()
```

file_hash_of_apk() → dict

Generates the MD5, SHA1 and SHA256 hashes of the APK.

Returns Returns dict containing MD5, SHA1 and SHA256 hash of APK.

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.file_hash_of_apk()
```

file_html_files() → list

Returns a list of found html files

Returns Array of HTML files

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_html_files()files
```

file_interesting() → list

Returns a list of found bks keystore files

Returns Array of interesting filetypes

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_interesting()
```

file_jar_files() → list

Returns a list of found jar files

Returns Array of JAR files

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_jar_files()files
```

file_js_files() → list

Returns a list of found js files

Returns Array of JS files

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_js_files()files
```

file_kivy_app() → bool

This method checks to see if the app is a Kivy compiled application. Kivy is a python framework for application development

Returns True if kivy app, else False

Return type bool

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_kivy_app()
```

file_native_code() → list

Returns a string of available native code compitability if present

Returns List of native code presence

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.file_native_code()
```

file_other_langs() → dict

Checks to see if any other frameworks is being used in this app

Returns Dict of other android development frameworks

Return type dict

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_other_langs()
```

file_react_app() → bool

This method checks to see if the app is developed using the Facebook React framework

Returns True if React app, else False

Return type bool

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_react_app()
```

file_res_strings() → list

This method looks enumerates the strings found in sources/res/values/strings.xml.

Returns Array of found strings

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_res_strings()
```

file_resource_xml() → list

Returns a list of found xml files from the resources directory. These files usually contains configuration options and may contain secrets.

Returns Array of resource xml files

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_resource_xml()files
```

file_shared_libs_file_paths() → list

This method enumerates for shared objects, and returns a list of their paths

Returns a list of database file paths

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_shared_libs_file_paths()
```

file_xml_files() → list

Returns a list of found xml files

Returns Array of XML files

Return type list

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.file_xml_files()files
```

manifest_activities() → list

Returns a list of all activities and all related attributes | Reference | Reference

Returns An array of all the activities from the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_activities()
```

manifest_activity_alias() → list

Returns a list of all activity-alias and all related attributes | Reference

Returns A list of aliased activies

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_activity_alias()
```

manifest_allow_backup() → bool

Returns true if the allow backup flag is set for the APK | Reference

Returns Returns true if backup is allowed. Else False

Return type bool

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_allow_backup()
```

manifest_android_version() → dict

Returns the version number matching for min and target sdk.

Returns Android versions based on min and target sdk

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_android_version()
```

manifest_application_node() → dict

Returns a dictionary of all values that are found in the application node | Reference

Returns A dictionary of the application node from the manifest

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_application_node()
```

manifest_bind_permissions() → list

Returns a list of permissions that have the BIND property. This allows this permission scope to be executed with the scope of the system

list List of BIND permissions

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_bind_permissions()
```

manifest_custom_permission() → list

Parses the manifest for permissions and returns a dict of only custom permissions. | Referene

Returns Custom permissions

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_custom_permission()
```

manifest_dangerous_permission() → list

Parses the manifest for permissions and returns a dict of only dangerous permissions | Reference Android SDK | Referene

Returns Dangerous permissions

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_dangerous_permission()
```

manifest_debuggable() → bool

Returns true if the debuggable flag is set for the APK | Reference | Reference | Reference

Returns Returns True if debuggable, else False

Return type bool

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_debuggable()
```

manifest_exported_providers() → list

Returns a list of all providers and all related attributes | Reference | Reference OWASP

Returns a list of exported provider nodes from the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_exported_providers()
```

manifest_intent_uri_filter() → list

Parses the manifest for permissions and returns a dict of only dangerous permissions | Referene

Returns Intent filter uri's

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_intent_uri_filter()
```

manifest_main_activity() → dict

Returns the main launchable activity as a dict

Returns Main activity and its attributes

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_main_activity()
```

manifest_meta_data() → list

Returns the contents inside meta-data nodes | Reference

Returns a list of meta-data nodes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_meta_data()
```

manifest_min_sdk() → int

Returns the minimum SDK from the APK | Reference

Returns Min SDK

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_min_sdk()
```

manifest_package_name() → str

Returns the package name of the APK | Reference

Returns Package name as a string

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_package_name()
```

manifest_permission(*merged: bool = True*) → list

Returns a list of application permission and their attributes | Reference

Parameters **merged** (*bool*) – Merge the two permisison types into one list. Defaults to True

Returns Permissions and their attributes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_permission()
```

manifest_platform_build_version_code() → int

Returns the platform build version code from the APK

Returns Platform version code

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_platform_build_version_code()
```

manifest_platform_build_version_name() → str

Returns the platform build version name from the APK

Returns Platform version name

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_platform_build_version_name()
```

manifest_providers() → list

Returns a list of all providers and all related attributes | Reference | Reference

Returns a list of registered providers in the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_providers()
```

manifest_receivers() → list

Returns a list of all receivers and all related attributes | Reference

Returns a list receivers registered in the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_receivers()
```

manifest_secrets() → list

Find all secrets hidden in AndroidManifest.xml like tokens, keys etc.

Returns a list of common secrets hardcoded in the manifest.

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAnroid('/path/to/apk')
>>> a.manifest_secrets()
```

manifest_services() → list

Returns a list of all services and all related attributes | Reference

Returns a list of registered services in the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_services()
```

manifest_signature_permission() → list

Parses the manifest for permissions and returns a dict of only signature permissions | Reference Android SDK | Referene

Returns Signature permissions

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_signature_permission()
```

manifest_target_sdk() → int

Returns the target SDK from the APK | Reference

Returns Target SDK number

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_target_sdk()
```

manifest_uses_configuration() → list

Returns the uses-configuration and all attributes from the APK | Reference

Returns uses configuration. Returns None if none found

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.uses_configuration()
```

manifest_uses_feature() → list

Returns a list of all uses-feature node. uses-feature is normally used to elaborate on permissions. | Reference

Returns Attributes of found uses-feature nodes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_uses_feature()
```

manifest_uses_library() → list

Returns the uses-library and all attributes from the APK | Reference

Returns uses library

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_uses_library()
```

manifest_uses_permission(*merged: bool = True*) → list

Returns a list of application permission and their attributes. This is the main way stating permissions in AndroidManifest.xml file | Reference

Parameters **merged** (*bool, optional*) – Merge the two permisison types into one list defaults to True

Returns Permissions and their attributes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_uses_permissions()
```

manifest_version_code() → int

Returns the version code from the APK | Reference

Returns Version code. None if not found

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_version_code()
```

manifest_version_name() → str

Returns the version name from the APK | Reference

Returns Version name from the manifest. None if not found

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_version_name()
```

other_ad_networks(*show_code=False*) → glorifiedgrep.out.GreppedOut

Show imports of the popular android ad networks. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_ad_networks()
```

other_all_urls(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_all_urls()
```

other_aws_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all AWS keys in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_aws_keys()
```

other_content_urlhandler(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all `content://` urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_content_urlhandler()
```

other_email_addresses(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find email addresses in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_email_addresses()
```

other_file_urlhandler(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all `file://` urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_file_urlhandler()
```

other_find_trackers_ads() → list

Find trackers included in the app. Currently it looks for 135 trackers.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns List of matched trackers

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_find_trackers_ads()
```

other_github_token(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all Github tokens in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_github_token()
```

other_google_ads_import(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find imports relevant to Google ads

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_google_ads_import()
```

other_http_urls (*show_code=False*) → glorifiedgrep.out.GreppedOut

Find HTTP urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_http_urls()
```

other_ip_address (*show_code=False*) → glorifiedgrep.out.GreppedOut

Find IP addresses in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_ip_address()
```

other_password_in_url (*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all passwords in urls. Usually used for basic authentication

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_password_in_url()
```

other_secret_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_secret_keys()
```

other_unicode_chars(*script: str = 'Hangul', show_code=False*)

Find unicode characters representing differnt character sets from different languages in the decompiled apk. Supports both Unicode Scripes and Unicode Blocks. See the reference for supported ranges. | Reference

Parameters

- **script** (*string, default Hangul*) – Any supported Unicode Script or Unicode Blocks. Ex: `Han` for Chinese characters.
- **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_chinese_chars()
```

other_websocket_urlhandler(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all `ws://` or `wss://` urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_websocket_urlhandler()
```

owasp_cloud_backup(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of BackupAgent and its variations in the decompiled code | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_cloud_backup()
```

owasp_code_check_permission(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_code_check_permission()
```

owasp_crypto_imports(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the Java cryptographic imports in decompiled code | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_crypto_imports()
```

owasp_crypto_primitives(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the cryptographic primitives of the most frequently used classes and interfaces in decompiled code | Reference | Reference | Reference CWE

Parameters

- **show_code** (*bool, optional*) –
- **show_code** – See the full line of code, defaults to False

Returns name, line number and match

Return type dict

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_crypto_primitives()
```

owasp_debug_code(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate StrictMode code in the decompiled code. This will indicate if dev checks are left behind in the app. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_debug_code()
```

owasp_encrypted_sql_db(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getWritableDatabase if a paramter is passed to this method. This could indicate hardcoded passwords. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_storage()
```

owasp_external_cache_dir(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getExternalCacheDir method usage. If the app is using the external cache dir. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_cache_dir()
```

owasp_external_storage(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getExternal method usage. This indicates sections of code where the external storage of the Android device is being interacted with. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_storage()
```

owasp_get_secret_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getSecretKey and getPrivateKey methods. | Reference | Reference | Reference Android SDK | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_get_secret_keys()
```

owasp_hardcoded_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate hardcoded encryption keys and bytes used by SecretKeySpec. The decompiled code should be inspected to find hardcoded keys. | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_hardcoded_keys()
```

owasp_insecure_fingerprint_auth(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate insecure .authenticate public method where the first parameter is null. This results in purely event driven authentication and is not secure. | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_insecure_fingerprint_auth()
```

owasp_insecure_random(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the weak Ranom Java class. SecureRandom should be used instead | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_insecure_random()
```

owasp_intent_parameter(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_intent_parameter()
```

owasp_keychain_password(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of store(OutputStream... to check for hardcoded passwords for keychains. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_keychain_password()
```

owasp_keystore_cert_pinning(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate keystore ssl pinning in decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_keystore_cert_pinning()
```

owasp_properly_signed(*show_code=False*) → glorifiedgrep.out.GreppedOut

Returns the command that can be used to check if an app is properly signed. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_properly_signed()
```

owasp_runtime_exception_handling(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_runtime_exception_handling()
```

owasp_ssl_no_hostname_verification(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of onReceivedSslError which amy indicate cases where SSL errors are being ingored by the application. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_ssl_no_hostname_verification()
```

owasp_webview_cert_pinning(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate SSL cert pinning in webviews. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_cert_pinning()
```

owasp_webview_loadurl(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate where webviews are loading content from. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_loadurl()
```

owasp_webview_native_function(*show_code=False*) → glorifiedgrep.out.GreppedOut

Identify addJavascriptInterface which will allow JS to access native Java functions. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_native_function()
```

owasp_webview_ssl_ignore(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of onReceivedSslError which amy indicate cases where SSL errors are being ingored by the application. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_ssl_ignore()
```

owasp_world_read_write_files(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate if shared preferences are world readable or world writeable | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_world_read_write_files()
```

search_methods(*regex: str*) → list

Search methods available through the GlorifiedAndroid class. This does not search for methods in any classes from the utils module.

Parameters **regex** (*str*) – regex to search for

Returns List of matching methods

Return type list

```
>>> GlorifiedAndroid(apk).search_methods('intent')
```

1.1.1 CertInfo class

class glorifiedgrep.android.**CertInfo**(*cert_path*)

This class is used for analyzing the certificate that an application is signed with. All the methods from this class is available in `GlorifiedAndroid` class, but can also be used by itself by passing the path to the certificate.

Examples

```
>>> from glorifiedgrep.android import CertInfo
>>> cert = CertInfo('/path/to/cert)
```

__init__(*cert_path*)

The __init__ method for the CertInfo class

Parameters **cert_path** (*str*) – Path to the CERT.RSA file

```
>>> c = CertInfo('/path/to/CERT.RSA')
>>> c.cert_public_key
```

all_cert_analysis()

Property runs all available checks in _CertAnalysis

Returns Dictionary of all cert analysis

Return type dict

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_manifest_analysis()
```

cert_bits() → int

Certificate bit

Returns Certificate bits

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_bits()
```

cert_certificate() → glorifiedgrep.out.GreppedOut

Returns a PEM encoded certificate

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_certificate()
```

cert_digest() → dict

Returns the digest hash in md5. sha1 and sha256

Returns Dictionary of hashes

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_digest()
```

cert_issuer() → glorifiedgrep.out.GreppedOut

The entity that verified the information and signed the certificate

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_issuer()
```

cert_public_key() → glorifiedgrep.out.GreppedOut

Get the public key from CERT.RSA

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_public_key()
```

cert_serial_number() → int

Used to uniquely identify the certificate within a CA's systems. In particular this is used to track revocation information

Returns Certificate serial number

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_serial_number()
```

cert_signature_algorithm() → str

The algorithm used to sign the public key certificate

Returns Algorithm used to create the certificate

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_signature_algorithm()
```

cert_subject() → list

The entity a certificate belongs to: a machine, an individual, or an organization.

Returns Dict of certificate subjects CN, O, C, ST, L, OU, Cn

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_subject()
```

cert_valid_dates() → dict

The that the certificate is valid before, after and if expired

Returns Dict of dates and if exipred

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_valid_dates()
```

cert_version() → int

The certificate version number

Returns Version number of the certificate

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.cert_version()
```

CHAPTER 2

CodeAnalysis class

class glorifiedgrep.android.**CodeAnalysis**(*source_path*)

This class can be used to perform code analysis checks against an already decompiled APK. This class also interits all the OWASP class methods.

__init__(*source_path*)

The __init__ method for the CertInfo class

Parameters **cert_path** (*str*) – Path to the CERT.RSA file

```
>>> c = CertInfo('/path/to/some/dir')
>>> c.code_dex_classloader()
```

all_owasp_analysis()

Property runs all available checks in _OwaspMasvs

Returns Dictionary of all other analysis

Return type dict

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_owasp_analysis()
```

code_accessibility_service(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses AccessibilityService and its various classes. It also looks for the accessibilityEvent method. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_accessibility_service()
```

code_add_javascriptinterface(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Leads to vulnerabilities in android version jellybean and below | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_add_javascriptinterface()
```

code_android_contacts_content_provider(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Idicates imports, or any other place where the ContactsContract class and its providors are being used. This typically indicates that the app can read various contact information from the phones contact list. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_android_contacts_content_provider()
```

code_apache_http_get_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpGet method from the apache library. This is generally used to make GET requests. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_get_request()
```

code_apache_http_other_request_methods(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpPut, HttpDelete, HttpHead, HttpTrace and HttpOptions methods from the apache library. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_apache_http_post_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpPost method from the apache library. This is generally used to make GET requests. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_api_builder(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method makes a best effort to detect api string builders within the decompiled Java code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_api_builder()
```

code_apk_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if calls to apk files are hardcoded.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apk_files()
```

code_aws_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where AWS queries are being made. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_aws_query()
```

code_base64_decode(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify base64 decode operations.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_base64_decode()
```

code_base64_encode(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify base64 encode operations.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_base64_encode()
```

code_boot_completed_persistance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses BOOT_COMPLETED action which is typically used to start a service or a receiver on reboot. This indicates persistance. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_boot_completed_persistance()
```

code_broadcast_messages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what broadcast messages are being sent in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_broadcast_messages()
```

code_broadcast_send(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify code that indicates broadcast messages being sent.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_broadcast_send()
```

code_browser_db_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the browser db. This db usually includes browsing history. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_browser_db_access()
```

code_byte_constants(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will create a dictionary of hardcoded byte constants.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_byte_constants()
```

code_call_log(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that retreives call logs. May be possible malware behaviour. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_call_log()
```

code_camera_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the camera and picture taking functionality. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_camera_access()
```

code_cipher_instance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of Cipher.getInstance in the decompiled source. class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_cipher_instance()
```

code_class_extends(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any classes that are extending another class.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_class_extends()
```

code_class_init(*class_name: str, show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will first identify import statemants from the provided `class_name` and then look for all new instances of `new class_name`. `class_name` can either be a class like Date, or a package name like java.utils.Date

Parameters

- **class_name** (*str*) – A valid class name. Can be either name; i.e. *Date*, or package name i.e *java.utils.Date*.
- **show_code** (*bool, optional*) – Show the full matched line, by default False, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_class_init()
```

code_clipboard_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where valies are being set or read from the clipboard. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_clipboard_manager()
```

code_command_exec(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all commands executed in shell using /bin/sh or .exec() in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_command_exec()
```

code_cookies(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where cookies are being set. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_cookies()
```

code_create_new_file(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that creates new files in the android system. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_new_file()
```

code_create_sockets(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

An InetSocketAddress is a special SocketAddress designed to represent the standard TCP Protocol address, so it thus has methods to set/query the host name, IP address, and Socket of the remote side of the connection (or, in fact the local side too) | Reference Android SDK | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_sockets()
```

code_create_tempfile(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all code which is using Java createTempFile | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_tempfile()
```

code_database_interaction(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is reading database files. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_database_interaction()
```

code_database_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that queries any database on the device. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_database_query()
```

code_debuggable_check(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for code what will check if the app is debuggable at run time. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_debuggable_check()
```

code_debugger_check (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for usage of isDebuggerConnected in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_debugger_check()
```

code_deserialization (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

ObjectInputSteam when used with 'readObject' 'readObjectNodData' 'readResolve' 'readExternal' will likely result in a Deserialization vulnerability | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_deserialization()
```

code_device_id (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where device id is being obtained. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_device_id()
```

code_device_serial_number(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for Build.SERIAL which can sometimes be used in addition with other things to build unique tokens. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_device_serial_number()
```

code_download_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses the DownloadManager class to download files from onlines services. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_download_manager()
```

code_dynamic_dexclassloader(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of DexClassLoader in the decompiled source. This can be used to execute code not installed as part of an application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_dynamic_dexclassloader()
```

code_dynamic_other_classloader(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of BaseDexClassLoader, SecureClassLoader, DelegateLastClassLoader, DexClass-

Loader, InMemoryDexClassLoader, PathClassLoader, URLClassLoader, Classloader in the decompiled source. This can be used to execute code not installed as part of an application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_dynamic_other_classloader()
```

code_exif_data (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects if the ExifInterface class is imported and then instantiated. This class is typically used to either set or get meta data from images | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_exif_data()
```

code_external_file_access (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where external files are being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_external_file_access()
```

code_file_observer (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of the FileObserver class being used. This class is used to check for file access or change and fire and event. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_file_observer()
```

code_file_read(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for FileInputStream within the decompiled Java code which would indicate which files the app is reading. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_file_read()
```

code_file_write(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for getByes() method which can indicate files being written by the app. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_write_file()
```

code_find_intents(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify intent builders.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_find_intents()
```

code_firebase_imports(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_firebase_imports()
```

code_get_environment_var(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for usage of getenv in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_get_environment_var()
```

code_google_api_keys(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Searches for Firebase or Google services API keys. It is likely that an app that uses Firebase will have keys in their sources, but these keys should be checked for what kind of access they allow.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_google_api_keys()
```

code_gps_location(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where GPS locations are being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_gps_location()
```

code_hashing_algorithms(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify hashing algorithms being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_hashing_algorithms()
```

code_hashing_custom(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify custom hashing algorithms being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_hashing_custom()
```

code_http_request_methods(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what HTTP request methods are being used. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_http_request_methods()
```

code_imports(*class_name: str*) → list

Returns an array of filepaths where a import statement matched the class_name. It does use a word boundary to get more of an exact match

Parameters **class_name** (*str*) – Name of the absolute or relative class

Returns List of file paths where a match has been found

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_imports()
```

code_intent_filters(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This identifies all the different types of intent filters

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_intent_filters()
```

code_intent_parameters(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify usage of the getStringExtra which is used to create parameters for intents. | Reference Android SDK | Reference OWASP

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_intent_parameters()
```

code_invisible_elements(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code will set the visibility of an element to invisible. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_invisible_elements()
```

code_jar_urlconnection(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is using the JarURLConnection API. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_jar_urlconnection()
```

code_js_read_file(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Gets or Sets whether JavaScript running in the context of a file scheme URL can access content from other file scheme URLs. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_js_read_file()
```

code_key_generator(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of KeyGenerator and its methods in the decompiled source. This class provides the functionality of a secret (symmetric) key generator | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_key_generator()
```

code_keystore_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where Bouncy castle bks or jks files are being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_keystore_files()
```

code_load_native_library(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method identifies where native libaries and loaded in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_load_native_library()
```

code_location(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
Identifies code that receives location information. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_location()
```

code_location_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
Identifies code that receives updated location information. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_location_manager()
```

code_logging(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method looks for the usage of Log class from Android SDK. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_logging()
```

code_make_http_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify when a HTTP connection is being made in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_make_http_request()
```

code_make_https_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify when a HTTPS connection is being made in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_make_http_request()
```

code_mediastore(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_mediastore()
```

code_notification_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that can access notifications. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_notification_access()
```

code_notification_manager (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that controls notifications. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_notification_manager()
```

code_null_cipher (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify nullciphers are being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_null_cipher()
```

code_object_deserialization (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where cookies are being set. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_object_deserialization()
```

code_package_installed(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the usage of the getInstalledPackages method from the PackageManager class. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_parse_uri(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is parsing a URI. This could be related to web urls, or content provider urls. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_parse_uri()
```

code_password_finder(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify possible passwords in the code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_password_finder()
```

code_phone_sensors(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that initiates various sensors available by Android. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_phone_sensors()
```

code_rabbit_amqp(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Checks if Rabbit amqp imports are present

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_rabbit_amqp()
```

code_read_sms_messages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Searches for SmsMessage class which is typically used to read SMS messages send to a device. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_read_sms_messages()
```

code_reflection(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that allows reflections in Java. This is a finding. Refer to the references for the risk and usage of reflections. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_reflection()
```

code_regex_matcher(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is processing regex. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_regex_matcher()
```

code_regex_pattern(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that compiles regex patterns. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_regex_pattern()
```

code_root_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that indicates if the app requests su access.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_root_access()
```

code_screenshots(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies usage of Bitmap and BitmapFactory classes. Although these are for bitmap compression and manipulation, they are often used to take screenshots. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_screenshots()
```

code_sdcard(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify strings matching sdcard usage.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sdcard()
```

code_search(*regex: str, rg_options: str = '', show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Run any checks against the decompiled code. The regex should be in raw string format

Parameters

- **regex** (*str*) – Regex pattern
- **rg_options** (*str*) – ripgrep options, space seperated string, defaults to ''
- **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

code_send_sms_text(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code can send SMS/Text messages. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_send_sms_text()
```

code_services(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what services are being started or being bound to. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_services()
```

code_shared_preferences (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method discovers SharePreference and getSharePreference from the decompiled code. Interface for accessing and modifying preference data returned by Context.getSharedPreferences within the decompiled Java code. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_shared_preferences()
```

code_sim_information (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where device sim card information is being obtained. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sim_information()
```

code_sql_injection_points (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for execquery. If user input is used in this query, this will lead to SQL injection. | Reference | Reference | Reference | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_injection_points()
```

code_sql_injection_user_input (*show_code=False*)

Find places in code where a variable is being concatanated with a SQL statement

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns

- *GreppedOut* – GreppedOut object
- *Examples*
- ——— (*rtype: dict*)
- *>>> from glorifiedgrep import GlorifiedAndroid*
- *>>> a = GlorifiedAndroid('/path/to/apk')*
- *>>> a.code_sql_inection_points()*

code_sql_java_implementation (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any other SQL queries that are implemented in Java. This searches for .query, .insert, .update and .delete methods. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_java_implementation()
```

code_sql_query_other (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any other SQL queries like INSERT, DROP etc in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_query_other()
```

code_sql_select_raw_query (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any SELECT queries in the decompiled code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_select_raw_query()
```

code_sqlcipher_password(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This getWritableDatabase and the getReadableDatabase methods from sqlcipher classes (3rd party) takes the db password as their argument. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sqlcipher_password()
```

code_sqlite_operations(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This getWritableDatabase and the getReadableDatabase methods db instances for sqlite opertations. These calls can be followed to check what data is being entered in the database. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sqlite_operations()
```

code_ssl_connections(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if SSL is being used by the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_ssl_connections()
```

code_stack_trace(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where AWS queries are being made. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_stack_trace()
```

code_static_iv(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify static IV's. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_static_iv()
```

code_string_constants(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will create a dictionary of hardcoded string constants.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_string_constants()
```

code_stub_packed(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for indication that the application is packed.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_stub_packed()
```

code_system_file_exists(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects if the exists method from the File class is being called. This method is typically used to check if the path in the class constructor exists in the system. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_system_service (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify systemservices being called. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_system_service()
```

code_tcp_sockets (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify TCP sockets being opened by the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_tcp_sockets()
```

code_trust_all_ssl (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that willl allow all SSL connections to succeed without verifying the hostname. This is a finding. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_trust_all_ssl()
```

code_udp_sockets(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify UDP sockets being opened by the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_udp_sockets()
```

code_weak_hashing(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where weak hashing algorithems such as MD5, MD4, SHA1 or any RC hashes are used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_weak_hashing()
```

code_websocket_usage(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects common Websockets init classes. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_websocket_usage()
```

code_webview_content_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where the webview has can access data from a content provider. | Reference Android SDK | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_content_access()
```

code_webview_database(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This allows developers to determine whether any WebView used in the application has stored any of the following types of browsing data and to clear any such stored data for all WebViews in the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_database()
```

code_webview_debug_enabled(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks to see if debug is enabled in webview. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_debug_enabled()
```

code_webview_file_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where the webview has file access. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_file_access()
```

code_webview_get_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify webview get requests. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_get_request()
```

code_webview_js_enabled(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method looks for any webview implementations where JavaScript is enabled. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_js_enabled()
```

code_webview_post_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify webview get requests. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_post_request()
```

code_xml_processor(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify possible weaknesses in XML parsing and creation. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xml_processor()
```

code_xor_encryption(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for XOR encryption operation within the decompiled code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xor_encryption()
```

code_xpath(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if SSL is being used by the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xpath()
```

owasp_cloud_backup(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of BackupAgent and its variations in the decompiled code | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_cloud_backup()
```

owasp_code_check_permission(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_code_check_permission()
```

owasp_crypto_imports(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the Java cryptographic imports in decompiled code | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_crypto_imports()
```

owasp_crypto_primitives(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the cryptographic primitives of the most frequently used classes and interfaces in decompiled code | Reference | Reference | Reference CWE

Parameters

- **show_code** (*bool, optional*) –
- **show_code** – See the full line of code, defaults to False

Returns name, line number and match

Return type dict

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_crypto_primitives()
```

owasp_debug_code(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate StrictMode code in the decompiled code. This will indicate if dev checks are left behind in the app. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_debug_code()
```

owasp_encrypted_sql_db(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getWritableDatabase if a paramter is passed to this method. This could indicate hardcoded passwords. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_storage()
```

owasp_external_cache_dir(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getExternalCacheDir method usage. If the app is using the external cache dir. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_cache_dir()
```

owasp_external_storage(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getExternal method usage. This indicates sections of code where the external storage of the Android device is being interacted with. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_storage()
```

owasp_get_secret_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getSecretKey and getPrivateKey methods. | Reference | Reference | Reference Android SDK | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_get_secret_keys()
```

owasp_hardcoded_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate hardcoded encryption keys and bytes used by SecretKeySpec. The decompiled code should be inspected to find hardcoded keys. | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_hardcoded_keys()
```

owasp_insecure_fingerprint_auth(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate insecure .authenticate public method where the first parameter is null. This results in purely event driven authentication and is not secure. | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_insecure_fingerprint_auth()
```

owasp_insecure_random(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the weak Ranom Java class. SecureRandom should be used instead | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_insecure_random()
```

owasp_intent_parameter(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_intent_parameter()
```

owasp_keychain_password(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of store(OutputStream... to check for hardcoded passwords for keychains. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_keychain_password()
```

owasp_keystore_cert_pinning(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate keystore ssl pinning in decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_keystore_cert_pinning()
```

owasp_properly_signed(*show_code=False*) → glorifiedgrep.out.GreppedOut

Returns the command that can be used to check if an app is properly signed. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_properly_signed()
```

owasp_runtime_exception_handling(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_runtime_exception_handling()
```

owasp_ssl_no_hostname_verification(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of onReceivedSslError which amy indicate cases where SSL errors are being ingored by the application. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_ssl_no_hostname_verification()
```

owasp_webview_cert_pinning(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate SSL cert pinning in webviews. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_cert_pinning()
```

owasp_webview_loadurl(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate where webviews are loading content from. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_loadurl()
```

owasp_webview_native_function(*show_code=False*) → glorifiedgrep.out.GreppedOut

Identify addJavascriptInterface which will allow JS to access native Java functions. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_native_function()
```

owasp_webview_ssl_ignore(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of onReceivedSslError which amy indicate cases where SSL errors are being ingored by the application. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_ssl_ignore()
```

owasp_world_read_write_files(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate if shared preferences are world readable or world writeable | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_world_read_write_files()
```

CHAPTER 3

ParseManifest class

class glorifiedgrep.android.**ParseManifest**(*manifest_path*)

This class can be used to just parse an AnroidManifest.xml file and parse it. This class does not decompile an APK file

__init__(*manifest_path*)

The __init__ method for the ParseManifest class

Parameters **manifest_path** (*str*) – Path to the manifest file

```
>>> a = ParseManifest('/path/to/AndroidManifest.xml')
>>> a.activities
```

all_manifest_analysis() → dict

Property runs all available checks in _ManifestAnalysis

Returns Dictionary of all analysis

Return type dict

manifest_activities() → list

Returns a list of all activities and all related attributes | Reference | Reference

Returns An array of all the activities from the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_activities()
```

manifest_activity_alias() → list

Returns a list of all activity-alias and all related attributes | Reference

Returns A list of aliased activies

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_activity_alias()
```

manifest_allow_backup() → bool

Returns true if the allow backup flag is set for the APK | Reference

Returns Returns true if backup is allowed. Else False

Return type bool

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_allow_backup()
```

manifest_android_version() → dict

Returns the version number matching for min and target sdk.

Returns Android versions based on min and target sdk

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_android_version()
```

manifest_application_node() → dict

Returns a dictionary of all values that are found in the application node | Reference

Returns A dictionary of the application node from the manifest

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_application_node()
```

manifest_bind_permissions() → list

Returns a list of permissions that have the BIND property. This allows this permission scope to be executed with the scope of the system

list List of BIND permissions

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_bind_permissions()
```

manifest_custom_permission() → list

Parses the manifest for permissions and returns a dict of only custom permissions. | Referene

Returns Custom permissions

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_custom_permission()
```

manifest_dangerous_permission() → list

Parses the manifest for permissions and returns a dict of only dangerous permissions | Reference Android SDK | Referene

Returns Dangerous permissions

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_dangerous_permission()
```

manifest_debuggable() → bool

Returns true if the debuggable flag is set for the APK | Reference | Reference | Reference

Returns Returns True if debuggable, else False

Return type bool

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_debuggable()
```

manifest_exported_providers() → list

Returns a list of all providers and all related attributes | Reference | Reference OWASP

Returns a list of exported provider nodes from the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_exported_providers()
```

manifest_intent_uri_filter() → list

Parses the manifest for permissions and returns a dict of only dangerous permissions | Referene

Returns Intent filter uri's

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_intent_uri_filter()
```

manifest_main_activity() → dict

Returns the main launchable activity as a dict

Returns Main activity and its attributes

Return type dict

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_main_activity()
```

manifest_meta_data() → list

Returns the contents inside meta-data nodes | Reference

Returns a list of meta-data nodes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_meta_data()
```

manifest_min_sdk() → int

Returns the minimum SDK from the APK | Reference

Returns Min SDK

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_min_sdk()
```

manifest_package_name() → str

Returns the package name of the APK | Reference

Returns Package name as a string

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_package_name()
```

manifest_permission(*merged: bool = True*) → list

Returns a list of application permission and their attributes | Reference

Parameters **merged** (*bool*) – Merge the two permisison types into one list. Defaults to True

Returns Permissions and their attributes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_permission()
```

manifest_platform_build_version_code() → int

Returns the platform build version code from the APK

Returns Platform version code

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_platform_build_version_code()
```

manifest_platform_build_version_name() → str

Returns the platform build version name from the APK

Returns Platform version name

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_platform_build_version_name()
```

manifest_providers() → list

Returns a list of all providers and all related attributes | Reference | Reference

Returns a list of registered providers in the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_providers()
```

manifest_receivers() → list

Returns a list of all receivers and all related attributes | Reference

Returns a list receivers registered in the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_receivers()
```

manifest_secrets() → list

Find all secrets hidden in AndroidManifest.xml like tokens, keys etc.

Returns a list of common secrets hardcoded in the manifest.

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAnroid('/path/to/apk')
>>> a.manifest_secrets()
```

manifest_services() → list

Returns a list of all services and all related attributes | Reference

Returns a list of registered services in the manifest

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_services()
```

manifest_signature_permission() → list

Parses the manifest for permissions and returns a dict of only signature permissions | Reference Android SDK | Referene

Returns Signature permissions

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_signature_permission()
```

manifest_target_sdk() → int

Returns the target SDK from the APK | Reference

Returns Target SDK number

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_target_sdk()
```

manifest_uses_configuration() → list

Returns the uses-configuration and all attributes from the APK | Reference

Returns uses configuration. Returns None if none found

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.uses_configuration()
```

manifest_uses_feature() → list

Returns a list of all uses-feature node. uses-feature is normally used to elaborate on permissions. | Reference

Returns Attributes of found uses-feature nodes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_uses_feature()
```

manifest_uses_library() → list

Returns the uses-library and all attributes from the APK | Reference

Returns uses library

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_uses_library()
```

manifest_uses_permission(*merged: bool = True*) → list

Returns a list of application permission and their attributes. This is the main way stating permissions in AndroidManifest.xml file | Reference

Parameters **merged** (*bool, optional*) – Merge the two permisison types into one list defaults to True

Returns Permissions and their attributes

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_uses_permissions()
```

manifest_version_code() → int

Returns the version code from the APK | Reference

Returns Version code. None if not found

Return type int

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_version_code()
```

manifest_version_name() → str

Returns the version name from the APK | Reference

Returns Version name from the manifest. None if not found

Return type str

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk)
>>> a.manifest_version_name()
```

CHAPTER 4

OWASPAnalysis class

class glorifiedgrep.android.**OWASPAnalysis**(*source_path*)

This class can be used to perform code analysis checks against an already decompiled APK

__init__(*source_path*)

The __init__ method for the CertInfo class

Parameters **cert_path** (*str*) – Path to the CERT.RSA file

```
>>> o = OWASPAnalysis('/path/to/some/dir')
>>> c.owasp_insecure_random()
```

all_owasp_analysis()

Property runs all available checks in _OwaspMasvs

Returns Dictionary of all other analysis

Return type dict

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_owasp_analysis()
```

owasp_cloud_backup(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of BackupAgent and its variations in the decompiled code | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_cloud_backup()
```

owasp_code_check_permission(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_code_check_permission()
```

owasp_crypto_imports(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the Java cryptographic imports in decompiled code | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_crypto_imports()
```

owasp_crypto_primitives(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the cryptographic primitives of the most frequently used classes and interfaces in decompiled code | Reference | Reference | Reference CWE

Parameters

- **show_code** (*bool, optional*) –
- **show_code** – See the full line of code, defaults to False

Returns name, line number and match

Return type dict

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_crypto_primitives()
```

owasp_debug_code(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate StrictMode code in the decompiled code. This will indicate if dev checks are left behind in the app. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_debug_code()
```

owasp_encrypted_sql_db(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getWritableDatabase if a paramter is passed to this method. This could indicate hardcoded passwords. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_storage()
```

owasp_external_cache_dir(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getExternalCacheDir method usage. If the app is using the external cache dir. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_cache_dir()
```

owasp_external_storage(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getExternal method usage. This indicates sections of code where the external storage of the Android device is being interacted with. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_external_storage()
```

owasp_get_secret_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of getSecretKey and getPrivateKey methods. | Reference | Reference | Reference Android SDK | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_get_secret_keys()
```

owasp_hardcoded_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate hardcoded encryption keys and bytes used by SecretKeySpec. The decompiled code should be inspected to find hardcoded keys. | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_hardcoded_keys()
```

owasp_insecure_fingerprint_auth(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate insecure .authenticate public method where the first parameter is null. This results in purely event driven authentication and is not secure. | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_insecure_fingerprint_auth()
```

owasp_insecure_random(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate uses of the weak Ranom Java class. SecureRandom should be used instead | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_insecure_random()
```

owasp_intent_parameter(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_intent_parameter()
```

owasp_keychain_password(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of store(OutputStream... to check for hardcoded passwords for keychains. | Reference | Reference | Reference Android SDK | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_keychain_password()
```

owasp_keystore_cert_pinning(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate keystore ssl pinning in decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_keystore_cert_pinning()
```

owasp_properly_signed(*show_code=False*) → glorifiedgrep.out.GreppedOut

Returns the command that can be used to check if an app is properly signed. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_properly_signed()
```

owasp_runtime_exception_handling(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate common exceptions thrown by RuntimeException from decompiled code. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_runtime_exception_handling()
```

owasp_ssl_no_hostname_verification(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of onReceivedSslError which amy indicate cases where SSL errors are being ingored by the application. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_ssl_no_hostname_verification()
```

owasp_webview_cert_pinning(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate SSL cert pinning in webviews. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_cert_pinning()
```

owasp_webview_loadurl(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate where webviews are loading content from. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_loadurl()
```

owasp_webview_native_function(*show_code=False*) → glorifiedgrep.out.GreppedOut

Identify addJavascriptInterface which will allow JS to access native Java functions. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_native_function()
```

owasp_webview_ssl_ignore(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate usage of onReceivedSslError which amy indicate cases where SSL errors are being ingored by the application. | Reference | Reference | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_webview_ssl_ignore()
```

owasp_world_read_write_files(*show_code=False*) → glorifiedgrep.out.GreppedOut

Locate if shared preferences are world readable or world writeable | Reference | Reference | Reference CWE

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.owasp_world_read_write_files()
```

CHAPTER 5

OtherAnalysis class

class glorifiedgrep.android.**OtherAnalysis**(*source_path*)

This calss can be used to gather arbitrary information like URL's, secret keys, tokens, chinese characters etc.

__init__(*source_path*)

The __init__ method for the OtherAnalysis class

Parameters **source_path** (*str*) – Path to folder where decompiled source code is

```
>>> o = OtherAnalysis('/path/to/some/dir')
>>> o.other_chinese_chars()
```

all_other_analysis()

Property runs all available checks in _OtherAnalysis

Returns Dictionary of all other analysis

Return type dict

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.all_other_analysis()
```

classmethod exodus_trackers(*trackers*)

Use this method to override the build in _TRACKERS constant with the response body from the exodus api. This is not recommended because some of the detection regex's from exodus are not valid. Example 'CrowdTangle': '.' The Exodus api url is https://reports.exodus-privacy.eu.org/api/trackers

Parameters **trackers** (*str*) – the json response body from the exodus api.

Examples

```
>>> import requests
>>> from glorifiedgrep.android.modules.constants import _Trackers
```

(continues on next page)

(continued from previous page)

```
>>> res = requests.get('https://reports.exodus-privacy.eu.org/api/trackers').
↪text
>>> _Trackers().exodus_trackers(res)
```

other_ad_networks(*show_code=False*) → glorifiedgrep.out.GreppedOut

Show imports of the popular android ad networks. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_ad_networks()
```

other_all_urls(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_all_urls()
```

other_aws_keys(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all AWS keys in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_aws_keys()
```

other_content_urlhandler(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all `content://` urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_content_urlhandler()
```

other_email_addresses(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find email addresses in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_email_addresses()
```

other_file_urlhandler(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all `file://` urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_file_urlhandler()
```

other_find_trackers_ads() → list

Find trackers included in the app. Currently it looks for 135 trackers.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns List of matched trackers

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_find_trackers_ads()
```

other_github_token(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all Github tokens in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_github_token()
```

other_google_ads_import(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find imports relevant to Google ads

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_google_ads_import()
```

other_http_urls(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find HTTP urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_http_urls()
```

other_ip_address(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find IP addresses in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_ip_address()
```

other_password_in_url(*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all passwords in urls. Usually used for basic authentication

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_password_in_url()
```

other_secret_keys (*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_secret_keys()
```

other_unicode_chars (*script: str = 'Hangul', show_code=False*)

Find unicode characters representing differnt character sets from different languages in the decompiled apk. Supports both Unicode Scripes and Unicode Blocks. See the reference for supported ranges. | Reference

Parameters

- **script** (*string, default Hangul*) – Any supported Unicode Script or Unicode Blocks. Ex: `Han` for Chinese characters.
- **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_chinese_chars()
```

other_websocket_urlhandler (*show_code=False*) → glorifiedgrep.out.GreppedOut

Find all `ws://` or `wss://` urls in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.other_websocket_urlhandler()
```

CHAPTER 6

MalwareBehaviour class

class glorifiedgrep.android.modules.malware.**MalwareBehaviour**(*apk_path: str = None, output_dir: str = None, project_dir: str = None, rg_path: str = 'rg', jadx_path: str = 'jadx', clean_dir: bool = False*)

This class is used to identify various behaviours that can be normal, but often displayed by malware. This class inherits from the _CodeAnalysis class directly, and is instantiated in the same manner as the GlorifiedAndroid class. Any parameters that the GlorifiedAndroid class takes can be passed to this class.

Parameters

- **apk_path** (*str*) – Path to the APK
- **output_dir** (*str*) – Output dir for decompilation and unzipping, defaults to /tmp/GlorifiedAndroid
- **project_dir** (*str*) – Project directory used for already decompiled and processed apks, defaults to None
- **json_output** (*bool*) – Returns a Json object instead of dict. Defaults to False
- **rg_path** (*str*) – path to ripgrep. Defaults to looking for it in path
- **jadx_path** (*str*) – path to jadx. Defaults to looking for it in path
- **clean_dir** (*bool*) – delete the output directory before processing

Raises

- **NotValidPythonVersion** – Raises if python version 3 is not used
- **DifferentAPKExists** – Raises if decompiled APK is different than what is already decompiled

- **DependentBinaryMissing** – Raises if ripgrep, or jadx is not found

```
>>> from glorifiedgrep.android.modules.malware import MalwareBehaviour
>>> m = MalwareBehaviour('/path/to/apk', output_dir='/out/dir')
```

__init__(*apk_path: str = None, output_dir: str = None, project_dir: str = None, rg_path: str = 'rg', jadx_path: str = 'jadx', clean_dir: bool = False*)

The init method for the whole GlorifiedAndroid module. This is interted throughout

Parameters

- **apk_path** (*str*) – Path to the APK
- **output_dir** (*str*) – Output dir for decompilation and unzipping, defaults to /tmp/glorified_android
- **project_dir** (*str*) – Project directory used for already decompiled and processed apks, defaults to None
- **rg_path** (*str*) – path to ripgrep. Defaults to looking for it in path
- **jadx_path** (*str*) – path to jadx. Defaults to looking for it in path
- **clean_dir** (*bool*) – delete the output directory before processing

Raises

- **NotValidPythonVersion** – Raises if python version 3 is not used
- **DifferentAPKExists** – Raises if decompiled APK is different than what is already decompiled
- **DependentBinaryMissing** – Raises if ripgrep, or jadx is not found

```
>>> # The default output directory is temp/GlorifiedAndroid folder. This can␣
↪be
>>> # overriden using output_dir='some/path'
>>> a = GlorifiedAndroid('/path/to/apk', output_dir='/out/dir')
```

Typically, the prefix for the file path is removed when processing filepaths in the various code analysis classes. This can be adjusted using

```
>>> a.remove_dir_prefix = ''
```

If **ripgrep** or **jadx** is not in path, analysis will not be complete. To pass a user defined path for either jadx or rg, the GlorifiedAndroid class can be instantiated as follows.

```
>>> a = GlorifiedAndroid('/path/to/apk', jadx_path='path/to/jadx', rg_path='/
↪path/to/rg')
```

code_accessibility_service(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses AccessibilityService and its various classes. It also looks for the accessibilityEvent method. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_accessibility_service()
```

code_add_javascriptinterface(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Leads to vulnerabilities in android version jellybean and below | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_add_javascriptinterface()
```

code_android_contacts_content_provider(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Idicates imports, or any other place where the ContactsContract class and its providors are being used. This typically indicates that the app can read various contact information from the phones contact list. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_android_contacts_content_provider()
```

code_apache_http_get_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpGet method from the apache library. This is generally used to make GET requests. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_get_request()
```

code_apache_http_other_request_methods(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpPut, HttpDelete, HttpHead, HttpTrace and HttpOptions methods from the apache library. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_apache_http_post_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the HttpPost method from the apache library. This is generally used to make GET requests. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_api_builder(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method makes a best effort to detect api string builders within the decompiled Java code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_api_builder()
```

code_apk_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if calls to apk files are hardcoded.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apk_files()
```

code_aws_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where AWS queries are being made. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_aws_query()
```

code_base64_decode(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify base64 decode operations.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_base64_decode()
```

code_base64_encode(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify base64 encode operations.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_base64_encode()
```

code_boot_completed_persistance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses BOOT_COMPLETED action which is typically used to start a service or a receiver on reboot. This indicates persistance. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_boot_completed_persistance()
```

code_broadcast_messages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what broadcast messages are being sent in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_broadcast_messages()
```

code_broadcast_send(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify code that indicates broadcast messages being sent.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_broadcast_send()
```

code_browser_db_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the browser db. This db usually includes browsing history. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_browser_db_access()
```

code_byte_constants(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will create a dictionary of hardcoded byte constants.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_byte_constants()
```

code_call_log(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that retreives call logs. May be possible malware behaviour. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_call_log()
```

code_camera_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the camera and picture taking functionality. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_camera_access()
```

code_cipher_instance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of Cipher.getInstance in the decompiled source. class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_cipher_instance()
```

code_class_extends(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any classes that are extending another class.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_class_extends()
```

code_class_init(*class_name: str, show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will first identify import statemants from the provided `class_name` and then look for all new instances of `new class_name`. `class_name` can either be a class like Date, or a package name like java.utils.Date

Parameters

- **class_name** (*str*) – A valid class name. Can be either name; i.e. *Date*, or package name i.e *java.utils.Date*.
- **show_code** (*bool, optional*) – Show the full matched line, by default False, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_class_init()
```

code_clipboard_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where valies are being set or read from the clipboard. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_clipboard_manager()
```

code_command_exec(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all commands executed in shell using /bin/sh or .exec() in the decompiled source

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_command_exec()
```

code_cookies(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where cookies are being set. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_cookies()
```

code_create_new_file(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that creates new files in the android system. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_new_file()
```

code_create_sockets(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

An InetSocketAddress is a special SocketAddress designed to represent the standard TCP Protocol address, so it thus has methods to set/query the host name, IP address, and Socket of the remote side of the connection (or, in fact the local side too) | Reference Android SDK | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_sockets()
```

code_create_tempfile(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all code which is using Java createTempFile | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_create_tempfile()
```

code_database_interaction(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is reading database files. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_database_interaction()
```

code_database_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that queries any database on the device. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_database_query()
```

code_debuggable_check(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for code what will check if the app is debuggable at run time. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_debuggable_check()
```

code_debugger_check (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for usage of isDebuggerConnected in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_debugger_check()
```

code_deserialization (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

ObjectInputSteam when used with 'readObject' 'readObjectNodData' 'readResolve' 'readExternal' will likely result in a Deserialization vulnerability | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_deserialization()
```

code_device_id (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where device id is being obtained. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_device_id()
```

code_device_serial_number(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for Build.SERIAL which can sometimes be used in addition with other things to build unique tokens. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_device_serial_number()
```

code_download_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses the DownloadManager class to download files from onlines services. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_download_manager()
```

code_dynamic_dexclassloader(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of DexClassLoader in the decompiled source. This can be used to execute code not installed as part of an application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_dynamic_dexclassloader()
```

code_dynamic_other_classloader(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of BaseDexClassLoader, SecureClassLoader, DelegateLastClassLoader, DexClass-

Loader, InMemoryDexClassLoader, PathClassLoader, URLClassLoader, Classloader in the decompiled source. This can be used to execute code not installed as part of an application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_dynamic_other_classloader()
```

code_exif_data (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects if the ExifInterface class is imported and then instantiated. This class is typically used to either set or get meta data from images | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_exif_data()
```

code_external_file_access (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where external files are being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_external_file_access()
```

code_file_observer (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of the FileObserver class being used. This class is used to check for file access or change and fire and event. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_file_observer()
```

code_file_read(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for FileInputStream within the decompiled Java code which would indicate which files the app is reading. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_file_read()
```

code_file_write(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for getByes() method which can indicate files being written by the app. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_write_file()
```

code_find_intents(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify intent builders.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_find_intents()
```

code_firebase_imports(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_firebase_imports()
```

code_get_environment_var(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for usage of getenv in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_get_environment_var()
```

code_google_api_keys(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Searches for Firebase or Google services API keys. It is likely that an app that uses Firebase will have keys in their sources, but these keys should be checked for what kind of access they allow.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_google_api_keys()
```

code_gps_location(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where GPS locations are being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_gps_location()
```

code_hashing_algorithms(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify hashing algorithms being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_hashing_algorithms()
```

code_hashing_custom(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify custom hashing algorithms being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_hashing_custom()
```

code_http_request_methods(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what HTTP request methods are being used. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_http_request_methods()
```

code_imports(*class_name: str*) → list

Returns an array of filepaths where a import statement matched the class_name. It does use a word boundary to get more of an exact match

Parameters **class_name** (*str*) – Name of the absolute or relative class

Returns List of file paths where a match has been found

Return type list

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_imports()
```

code_intent_filters(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This identifies all the different types of intent filters

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_intent_filters()
```

code_intent_parameters(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify usage of the getStringExtra which is used to create parameters for intents. | Reference Android SDK | Reference OWASP

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_intent_parameters()
```

code_invisible_elements(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code will set the visibility of an element to invisible. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_invisible_elements()
```

code_jar_urlconnection(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is using the JarURLConnection API. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_jar_urlconnection()
```

code_js_read_file(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Gets or Sets whether JavaScript running in the context of a file scheme URL can access content from other file scheme URLs. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_js_read_file()
```

code_key_generator(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find all instances of KeyGenerator and its methods in the decompiled source. This class provides the functionality of a secret (symmetric) key generator | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_key_generator()
```

code_keystore_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where Bouncy castle bks or jks files are being used.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_keystore_files()
```

code_load_native_library(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method identifies where native libaries and loaded in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_load_native_library()
```

code_location(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that receives location information. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_location()
```

code_location_manager(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that receives updated location information. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_location_manager()
```

code_logging(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for the usage of Log class from Android SDK. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_logging()
```

code_make_http_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify when a HTTP connection is being made in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_make_http_request()
```

code_make_https_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify when a HTTPS connection is being made in the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_make_http_request()
```

code_mediastore(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_mediastore()
```

code_notification_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that can access notifications. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_notification_access()
```

code_notification_manager (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that controls notifications. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_notification_manager()
```

code_null_cipher (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify nullciphers are being used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_null_cipher()
```

code_object_deserialization (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where cookies are being set. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_object_deserialization()
```

code_package_installed(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects the usage of the getInstalledPackages method from the PackageManager class. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_parse_uri(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is parsing a URI. This could be related to web urls, or content provider urls. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_parse_uri()
```

code_password_finder(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify possible passwords in the code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_password_finder()
```

code_phone_sensors(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that initiates various sensors available by Android. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_phone_sensors()
```

code_rabbit_amqp(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Checks if Rabbit amqp imports are present

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_rabbit_amqp()
```

code_read_sms_messages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Searches for SmsMessage class which is typically used to read SMS messages send to a device. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_read_sms_messages()
```

code_reflection(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that allows reflections in Java. This is a finding. Refer to the references for the risk and usage of reflections. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_reflection()
```

code_regex_matcher(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that is processing regex. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_regex_matcher()
```

code_regex_pattern(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that compiles regex patterns. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_regex_pattern()
```

code_root_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that indicates if the app requests su access.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_root_access()
```

code_screenshots(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies usage of Bitmap and BitmapFactory classes. Although these are for bitmap compression and manipulation, they are often used to take screenshots. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_screenshots()
```

code_sdcard(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify strings matching sdcard usage.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sdcard()
```

code_search(*regex: str, rg_options: str = '', show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Run any checks against the decompiled code. The regex should be in raw string format

Parameters

- **regex** (*str*) – Regex pattern
- **rg_options** (*str*) – ripgrep options, space seperated string, defaults to ''
- **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

code_send_sms_text(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code can send SMS/Text messages. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_send_sms_text()
```

code_services(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify what services are being started or being bound to. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_services()
```

code_shared_preferences(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method discovers SharePreference and getSharePreference from the decompiled code. Interface for accessing and modifying preference data returned by Context.getSharedPreferences within the decompiled Java code. | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_shared_preferences()
```

code_sim_information(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where device sim card information is being obtained. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sim_information()
```

code_sql_injection_points(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for execquery. If user input is used in this query, this will lead to SQL injection. | Reference | Reference | Reference | Reference | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_injection_points()
```

code_sql_injection_user_input(*show_code=False*)

Find places in code where a variable is being concatanated with a SQL statement

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns

- *GreppedOut* – GreppedOut object
- *Examples*
- ——— (*rtype: dict*)
- *>>> from glorifiedgrep import GlorifiedAndroid*
- *>>> a = GlorifiedAndroid('/path/to/apk')*
- *>>> a.code_sql_inection_points()*

code_sql_java_implementation (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any other SQL queries that are implemented in Java. This searches for .query, .insert, .update and .delete methods. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_java_implementation()
```

code_sql_query_other (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any other SQL queries like INSERT, DROP etc in the decompiled code. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_query_other()
```

code_sql_select_raw_query (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any SELECT queries in the decompiled code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sql_select_raw_query()
```

code_sqlcipher_password(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This getWritableDatabase and the getReadableDatabase methods from sqlcipher classes (3rd party) takes the db password as their argument. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sqlcipher_password()
```

code_sqlite_operations(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This getWritableDatabase and the getReadableDatabase methods db instances for sqlite opertations. These calls can be followed to check what data is being entered in the database. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_sqlite_operations()
```

code_ssl_connections(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if SSL is being used by the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_ssl_connections()
```

code_stack_trace(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where AWS queries are being made. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_stack_trace()
```

code_static_iv(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify static IV's. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_static_iv()
```

code_string_constants(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will create a dictionary of hardcoded string constants.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_string_constants()
```

code_stub_packed(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for indication that the application is packed.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_stub_packed()
```

code_system_file_exists(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects if the exists method from the File class is being called. This method is typically used to check if the path in the class constructor exists in the system. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_apache_http_post_request()
```

code_system_service (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify systemservices being called. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_system_service()
```

code_tcp_sockets (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify TCP sockets being opened by the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_tcp_sockets()
```

code_trust_all_ssl (*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that willl allow all SSL connections to succeed without verifying the hostname. This is a finding. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_trust_all_ssl()
```

code_udp_sockets(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify UDP sockets being opened by the decompiled code. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_udp_sockets()
```

code_weak_hashing(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify where weak hashing algorithems such as MD5, MD4, SHA1 or any RC hashes are used. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_weak_hashing()
```

code_websocket_usage(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Detects common Websockets init classes. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_websocket_usage()
```

code_webview_content_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where the webview has can access data from a content provider. | Reference Android SDK | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_content_access()
```

code_webview_database(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This allows developers to determine whether any WebView used in the application has stored any of the following types of browsing data and to clear any such stored data for all WebViews in the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_database()
```

code_webview_debug_enabled(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks to see if debug is enabled in webview. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_debug_enabled()
```

code_webview_file_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for any webview implementations where the webview has file access. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_file_access()
```

code_webview_get_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify webview get requests. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_get_request()
```

code_webview_js_enabled(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method looks for any webview implementations where JavaScript is enabled. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_js_enabled()
```

code_webview_post_request(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify webview get requests. | Reference Android SDK

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_webview_post_request()
```

code_xml_processor(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut
This method will identify possible weaknesses in XML parsing and creation. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xml_processor()
```

code_xor_encryption(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method looks for XOR encryption operation within the decompiled code.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xor_encryption()
```

code_xpath(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

This method will identify if SSL is being used by the application. | Reference

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> from glorifiedgrep import GlorifiedAndroid
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.code_xpath()
```

malware_access_call_logs(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identify classes commonly used with taking screenshots

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_access_call_logs()
```

malware_access_camera(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identify classes commonly used with accessing the camera.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_access_camera()
```

malware_accessibility_services(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses varios classes and methods related to accessibility services. Malware will often use this to have a higher level control of the device.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_accessibility_services()
```

malware_boot_completed_persistance(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses BOOT_COMPLETED action which is typically used to start a service or a receiver on reboot. This indicates persistance.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_boot_completed_persistance()
```

malware_browser_db_access(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that accesses the browser db. This db usually includes browsing history.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_browser_db_access()
```

malware_database_query(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies code that queries any database on the device.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_database_query()
```

malware_debug(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the app is either debuggable, or if it is connected to a debugger.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_debug()
```

malware_download_files(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the application uses the DownloadManager class to download files from onlines services.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_download_files()
```

malware_get_external_storage(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identify code that is commonly used to get path to the external storage directory.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_get_external_storage()
```

malware_get_installed_packages(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if the getInstalledPackages method from the PackageManager class is being called. Malware will usually use this method to enumerate all the installed apps in a device.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_obtain_file_metadata()
```

malware_obtain_file_metadata(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identifies if he MediaStore class or some of its common subclasses are being used by the app. These classes are used to get media file metadata from both internal and external storage.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_obtain_file_metadata()
```

malware_screen_unlock(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Find android.intent.action.USER_PRESENT in the manifest which is usually an intent used to detect when the screen is unlocked. The receiver for the intent should be inspected more closely.

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> a = GlorifiedAndroid('/path/to/apk')
>>> a.malware_screen_unlock()
```

malware_take_screenshots(*show_code: bool = False*) → glorifiedgrep.out.GreppedOut

Identify classes commonly used with taking screenshots

Parameters **show_code** (*bool, optional*) – Show the full matched line, by default False

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> m.malware_take_screenshots()
```

CHAPTER 7

Utils class

7.1 JKS class

class glorifiedgrep.android.modules.utils.**JKS**(*jks_file: str, jks_password: str*)

Process and get various information from jks files

Parameters

- **str** (*jks_password*) – File path to jks file
- **str** – password to the jks file

```
>>> from glorifiedgrep.android.modules.utils import JKS
>>> j = JKS('/path/to/file', 'secretpassword')
```

__init__(*jks_file: str, jks_password: str*)

The init function of the JKS class

Parameters

- **jks_file** (*str*) – The path to the .jks file
- **jks_password** (*str*) – The password for the jks file

jks_certificate() → list

Get the certificate from the jks file

Returns jks certificates

Return type list

Examples

```
>>> j.jks_certificate()
```

jks_key_alias() → list
Get the keystore alias from jks file

Returns jks keystore aliases

Return type list

Examples

```
>>> j.jks_key_alias()
```

jks_private_key() → list
Get the private key from jks files

Returns jks private keys if password is correct

Return type list

Examples

```
>>> j.jks_private_key()
```

7.2 BKS class

class glorifiedgrep.android.modules.utils.**BKS**(*bks_file*, *bks_password*)
Process and get various information from bks files

Parameters

- **str** (*bks_password*) – File path to bks file
- **str** – password to the bks file

```
>>> from glorifiedgrep.android.modules.utils import BKS
>>> b = BKS('/path/to/file', 'secretpassword')
```

__init__(*bks_file*, *bks_password*)
Initialize self. See help(type(self)) for accurate signature.

bks_certificate() → list
Prints the certificate from the bks file

Returns bks certificates

Return type list

Examples

```
>>> b.bks_certificate()
```

bks_keystore_alias() → list
Prints the keystore alias of the bks file

Returns bks keystore aliases

Return type list

Examples

```
>>> b.bks_keystore_alias()
```

bks_keystore_type() → list

Prints the keystore type of the bks file

Returns bks keystore type

Return type list

Examples

```
>>> b.bks_keystore_type()
```

7.3 NativeELFAnalysis class

class glorifiedgrep.android.modules.utils.**NativeELFAnalysis**(*elf_path: str*)

Class is used to handle the processing and analysis of native libraries included in the APK. It relies of lief to handle the processing. To install lief for py 3.7, follow instructions at https://github.com/lief-project/LIEF/issues/214

Parameters **str**(*elf_path*) – path to the lib file

```
>>> from glorifiedgrep.android.modules.utils import NativeELFAnalysis
>>> n = NativeELFAnalysis('/path/to/file.so')
```

__init__(*elf_path: str*)

Initialize self. See help(type(self)) for accurate signature.

elf_exported_symbols() → list

Returns a list of exported symbols from the binary

Returns Array of exports from the binary

Return type list

Examples

```
>>> n.elf_exported_symbols()
```

elf_header_info() → lief._pylief.ELF.Header

Returns a lief header object with information obtained from the binaries header

Returns **_pylief.ELF.Header** – Header object object

Return type object

Examples

```
>>> n.elf_header_info()
```

elf_imported_symbols() → list
Returns a list of imported symbols from the binary

Returns list of imports from the binary

Return type list

Examples

```
>>> n.elf_imported_symbols()
```

elf_libraries_binary() → list
Returns a list of libraries the binary is linked with

Returns Liked libraries

Return type list

Examples

```
>>> n.elf_libraries_binary()
```

elf_strings_from_binary() → list
Returns a list of strings from the binary

Returns Array of strings from the binary

Return type list

Examples

```
>>> n.elf_strings_from_binary()
```

7.4 NativeDEXAnalysis class

class glorifiedgrep.android.modules.utils.**NativeDEXAnalysis**(*dex_path: str*)
Class is used to handle the processing and analysis of dex files obtained from unzipping an APK. It relies of lief to handle the processing. To install lief for py 3.7, follow instructions at https://github.com/lief-project/LIEF/issues/214

Parameters **str** (*dex_path*) – path to the lib file

```
>>> from glorifiedgrep.android.modules.utils import NativeELFAnalysis
>>> n = NativeDEXAnalysis('/path/to/classes.dex')
```

__init__(*dex_path: str*)
This class analyzes native dex files that are not decompiled

Parameters **dex_path** (*str*) – Path to dex file

dex_classes() → Iterable[dict]
Parse the dex file and returns a list of class names and other information

Returns Returns a generator of dictionaries containing the name, full_name, package_name source_file, and method keys

Return type Iteratable

Examples

```
>>> n.dex_dex_classes()
```

dex_info() → Iterable[lief._pylief.DEX.File.classes]
Parse the dex file and returns a lief dex file object

Returns Returns a generator of containing the class names and their associated methods

Return type Iteratable

Examples

```
>>> n.dex_dex_info()
```

dex_methods() → Iterable[dict]
Parse the dex file and returns a dictionary of method information

Returns Returns a generator of dictionaries containing the name, class, parameters and return_type keys

Return type Iteratable

Examples

```
>>> n.dex_dex_methods()
```

dex_parse() → lief._pylief.DEX.File
Parse the dex file and returns a lief dex file object

Returns GreppedOut object

Return type *GreppedOut*

Examples

```
>>> n.dex_parse()
```

dex_strings() → Iterable[list]
Parse the dex file and returns a generator of string values

Returns Returns a generator of strings

Return type Iteratable

Examples

```
>>> n.dex_dex_strings()
```

7.5 SQL class

class glorifiedgrep.android.modules.utils.**SQL**(*db_path: str*)

Class is used to process, and extract various information from sqlite3 db files. It uses python sqlite3 standard library.

Parameters **str** (*db_path*) – path to the db file

```
>>> from glorifiedgrep.android.modules.utils import SQL
>>> s = SQL('/path/to/sql_db')
```

__init__(*db_path: str*)

The init method for the SQL class

Parameters **db_path** (*str*) – Path to a valid sqlite3 database file

sqldb_dump_database() → list

Dumps a list of all sql commands. Similar to `sqlite3 file.db .dump`

Returns An array of all dumped data

Return type list

Examples

```
>>> s.sqldb_dump_database()
```

sqldb_table_column_names(*table_name: str*) → list

Get all the column names for the specified table.

Parameters **table_name** (*str*) – An existing table name

Returns A list of column names from the specified table

Return type list

Examples

```
>>> s.sqldb_table_column_names()
```

sqldb_table_data(*table_name: str*) → list

Get all the data from the specified table.

Parameters **table_name** (*str*) – An existing table name

Returns Dumps an arry of table data

Return type list

Examples

```
>>> s.sqldb_table_data()
```

sqldb_tables() → list

Get all the table names from the db file

Returns A list of table names

Return type list

Examples

```
>>> s.sqldb_tables()
```

7.6 Utils class

class glorifiedgrep.android.modules.utils.**Utils**

General class for helpful utilities while working with unzipped or decompiled files

```
>>> from glorifiedgrep.android.modules.utils import Utils
>>> u = Utils()
```

__init__()

The init method for the whole GlorifiedAndroid module. This is interted throughout

Parameters

- **apk_path** (*str*) – Path to the APK
- **output_dir** (*str*) – Output dir for decompilation and unzipping, defaults to /tmp/glorified_android
- **project_dir** (*str*) – Project directory used for already decompiled and processed apks, defaults to None
- **rg_path** (*str*) – path to ripgrep. Defaults to looking for it in path
- **jadx_path** (*str*) – path to jadx. Defaults to looking for it in path
- **clean_dir** (*bool*) – delete the output directory before processing

Raises

- **NotValidPythonVersion** – Raises if python version 3 is not used
- **DifferentAPKExists** – Raises if decompiled APK is different than what is already decompiled
- **DependentBinaryMissing** – Raises if ripgrep, or jadx is not found

```
>>> # The default output directory is temp/GlorifiedAndroid folder. This can␣
↪be
>>> # overriden using output_dir='some/path'
>>> a = GlorifiedAndroid('/path/to/apk', output_dir='/out/dir')
```

Typically, the prefix for the file path is removed when processing filepaths in the various code analysis classes. This can be adjusted using

```
>>> a.remove_dir_prefix = ''
```

If **ripgrep** or **jadx** is not in path, analysis will not be complete. To pass a user defined path for either jadx or rg, the GlorifiedAndroid class can be instantiated as follows.

```
>>> a = GlorifiedAndroid('/path/to/apk', jadx_path='path/to/jadx', rg_path='/
↪path/to/rg')
```

jks_password_bruteforce (*jks_file: str*, *word_list: str*) → str
Bruteforce the password for a JKS keystore

Parameters

- **jks_file** (*str*) – Path to JKS keystore
- **word_list** (*str*) – Path to wordlist

Returns Valid password if found. Else False

Return type str

utils_xml_to_dict (*file_path: str*) → dict
Parse xml file and return as a dict object

Parameters **file_path** (*str*) – Path to a valid XML file

Returns A dictionary object representing the xml file

Return type list

Examples

```
>>> u.utils_xml_to_dict('/path/to/file.xml)
```

CHAPTER 8

GreppedOut class

class glorifiedgrep.out.**GreppedOut**(*data*)

Thre GreppedOut class is generally used to capture the output of code analysis methods and offers various helper attributes and properties.

Returns

Return type object

__init__(*data*)

Initialize self. See help(type(self)) for accurate signature.

count

A count of the number of items either in the array or dict that is returned. This is a property.

Returns Count of items

Return type int

exclude_file(*path: str*) → glorifiedgrep.out.GreppedOut

Exclude matches from the files which partially matches the path argument. This method can be chained for multiple file paths.

Returns GreppedOut object

Return type *GreppedOut*

files

Get a set of file names where matches were found

Returns Set of filenames

Return type set

in_file(*path: str*) → glorifiedgrep.out.GreppedOut

Only include matches from the files which partially matches the path argument. This method can be chained for multiple file paths.

Returns GreppedOut object

Return type *GreppedOut*

matches

Get only the code matches in an array

Returns List of matches

Return type list

CHAPTER 9

Resources

- Exodus privacy
- Android SDK
- OWASP MSTG
- CMU Coding standarsd
- Qark
- MobSF
- JSSEC Secure Android Coding

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Index

Symbols

__init__() (*glorifiedgrep.GlorifiedAndroid method*), 1
__init__() (*glorifiedgrep.android.CertInfo method*), 59
__init__() (*glorifiedgrep.android.CodeAnalysis method*), 63
__init__() (*glorifiedgrep.android.OWASPAnalysis method*), 113
__init__() (*glorifiedgrep.android.OtherAnalysis method*), 121
__init__() (*glorifiedgrep.android.ParseManifest method*), 103
__init__() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 128
__init__() (*glorifiedgrep.android.modules.utils.BKS method*), 166
__init__() (*glorifiedgrep.android.modules.utils.JKS method*), 165
__init__() (*glorifiedgrep.android.modules.utils.NativeDEXAnalysis method*), 168
__init__() (*glorifiedgrep.android.modules.utils.NativeELFAnalysis method*), 167
__init__() (*glorifiedgrep.android.modules.utils.SQL method*), 170
__init__() (*glorifiedgrep.android.modules.utils.Utils method*), 171
__init__() (*glorifiedgrep.out.GreppedOut method*), 173

A

all_cert_analysis() (*glorifiedgrep.android.CertInfo method*), 60
all_cert_analysis() (*glorifiedgrep.GlorifiedAndroid method*), 2
all_file_analysis() (*glorifiedgrep.GlorifiedAndroid method*), 2
all_manifest_analysis() (*glorifiedgrep.android.ParseManifest method*), 103
all_manifest_analysis() (*glorifiedgrep.GlorifiedAndroid method*), 2
all_other_analysis() (*glorifiedgrep.android.OtherAnalysis method*), 121
all_other_analysis() (*glorifiedgrep.GlorifiedAndroid method*), 2
all_owasp_analysis() (*glorifiedgrep.android.CodeAnalysis method*), 63
all_owasp_analysis() (*glorifiedgrep.android.OWASPAnalysis method*), 113
all_owasp_analysis() (*glorifiedgrep.GlorifiedAndroid method*), 2

B

BKS (*class in glorifiedgrep.android.modules.utils*), 166
bks_certificate() (*glorifiedgrep.android.modules.utils.BKS method*), 166
bks_keystore_alias() (*glorifiedgrep.android.modules.utils.BKS method*), 166
bks_keystore_type() (*glorifiedgrep.android.modules.utils.BKS method*), 167

C

cert_bits() (*glorifiedgrep.android.CertInfo method*), 60
cert_bits() (*glorifiedgrep.GlorifiedAndroid method*), 3
cert_certificate() (*glorifiedgrep.android.CertInfo method*), 60
cert_certificate() (*glorifiedgrep.GlorifiedAndroid method*), 3
cert_digest() (*glorifiedgrep.android.CertInfo method*), 60

cert_digest() (*glorifiedgrep.GlorifiedAndroid method*), 3
cert_issuer() (*glorifiedgrep.android.CertInfo method*), 60
cert_issuer() (*glorifiedgrep.GlorifiedAndroid method*), 3
cert_public_key() (*glorifiedgrep.android.CertInfo method*), 61
cert_public_key() (*glorifiedgrep.GlorifiedAndroid method*), 4
cert_serial_number() (*glorifiedgrep.android.CertInfo method*), 61
cert_serial_number() (*glorifiedgrep.GlorifiedAndroid method*), 4
cert_signature_algorithm() (*glorifiedgrep.android.CertInfo method*), 61
cert_signature_algorithm() (*glorifiedgrep.GlorifiedAndroid method*), 4
cert_subject() (*glorifiedgrep.android.CertInfo method*), 61
cert_subject() (*glorifiedgrep.GlorifiedAndroid method*), 4
cert_valid_dates() (*glorifiedgrep.android.CertInfo method*), 62
cert_valid_dates() (*glorifiedgrep.GlorifiedAndroid method*), 5
cert_version() (*glorifiedgrep.android.CertInfo method*), 62
cert_version() (*glorifiedgrep.GlorifiedAndroid method*), 5
CertInfo (*class in glorifiedgrep.android*), 59
code_accessibility_service() (*glorifiedgrep.android.CodeAnalysis method*), 63
code_accessibility_service() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 128
code_accessibility_service() (*glorifiedgrep.GlorifiedAndroid method*), 5
code_add_javascriptinterface() (*glorifiedgrep.android.CodeAnalysis method*), 64
code_add_javascriptinterface() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 129
code_add_javascriptinterface() (*glorifiedgrep.GlorifiedAndroid method*), 5
code_android_contacts_content_provider() (*glorifiedgrep.android.CodeAnalysis method*), 64
code_android_contacts_content_provider() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 129
code_android_contacts_content_provider() (*glorifiedgrep.GlorifiedAndroid method*), 6
code_apache_http_get_request() (*glorifiedgrep.android.CodeAnalysis method*), 64
code_apache_http_get_request() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 129
code_apache_http_get_request() (*glorifiedgrep.GlorifiedAndroid method*), 6
code_apache_http_other_request_methods() (*glorifiedgrep.android.CodeAnalysis method*), 64
code_apache_http_other_request_methods() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 129
code_apache_http_other_request_methods() (*glorifiedgrep.GlorifiedAndroid method*), 6
code_apache_http_post_request() (*glorifiedgrep.android.CodeAnalysis method*), 65
code_apache_http_post_request() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 130
code_apache_http_post_request() (*glorifiedgrep.GlorifiedAndroid method*), 6
code_api_builder() (*glorifiedgrep.android.CodeAnalysis method*), 65
code_api_builder() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 130
code_api_builder() (*glorifiedgrep.GlorifiedAndroid method*), 7
code_apk_files() (*glorifiedgrep.android.CodeAnalysis method*), 65
code_apk_files() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 130
code_apk_files() (*glorifiedgrep.GlorifiedAndroid method*), 7
code_aws_query() (*glorifiedgrep.android.CodeAnalysis method*), 66
code_aws_query() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 131
code_aws_query() (*glorifiedgrep.GlorifiedAndroid method*), 7
code_base64_decode() (*glorifiedgrep.android.CodeAnalysis method*), 66
code_base64_decode() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 131
code_base64_decode() (*glorifiedgrep.GlorifiedAndroid method*), 8
code_base64_encode() (*glorifiedgrep.android.CodeAnalysis method*), 66
code_base64_encode() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 131

code_base64_encode() (*glorifiedgrep.GlorifiedAndroid method*), 8
code_boot_completed_persistance() (*glorifiedgrep.android.CodeAnalysis method*), 66
code_boot_completed_persistance() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 131
code_boot_completed_persistance() (*glorifiedgrep.GlorifiedAndroid method*), 8
code_broadcast_messages() (*glorifiedgrep.android.CodeAnalysis method*), 67
code_broadcast_messages() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 132
code_broadcast_messages() (*glorifiedgrep.GlorifiedAndroid method*), 8
code_broadcast_send() (*glorifiedgrep.android.CodeAnalysis method*), 67
code_broadcast_send() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 132
code_broadcast_send() (*glorifiedgrep.GlorifiedAndroid method*), 9
code_browser_db_access() (*glorifiedgrep.android.CodeAnalysis method*), 67
code_browser_db_access() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 132
code_browser_db_access() (*glorifiedgrep.GlorifiedAndroid method*), 9
code_byte_constants() (*glorifiedgrep.android.CodeAnalysis method*), 67
code_byte_constants() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 132
code_byte_constants() (*glorifiedgrep.GlorifiedAndroid method*), 9
code_call_log() (*glorifiedgrep.android.CodeAnalysis method*), 68
code_call_log() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 133
code_call_log() (*glorifiedgrep.GlorifiedAndroid method*), 9
code_camera_access() (*glorifiedgrep.android.CodeAnalysis method*), 68
code_camera_access() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 133
code_camera_access() (*glorifiedgrep.GlorifiedAndroid method*), 10
code_cipher_instance() (*glorifiedgrep.android.CodeAnalysis method*), 68
code_cipher_instance() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 133
code_cipher_instance() (*glorifiedgrep.GlorifiedAndroid method*), 10
code_class_extends() (*glorifiedgrep.android.CodeAnalysis method*), 69
code_class_extends() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 134
code_class_extends() (*glorifiedgrep.GlorifiedAndroid method*), 10
code_class_init() (*glorifiedgrep.android.CodeAnalysis method*), 69
code_class_init() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 134
code_class_init() (*glorifiedgrep.GlorifiedAndroid method*), 11
code_clipboard_manager() (*glorifiedgrep.android.CodeAnalysis method*), 69
code_clipboard_manager() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 134
code_clipboard_manager() (*glorifiedgrep.GlorifiedAndroid method*), 11
code_command_exec() (*glorifiedgrep.android.CodeAnalysis method*), 70
code_command_exec() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 135
code_command_exec() (*glorifiedgrep.GlorifiedAndroid method*), 11
code_cookies() (*glorifiedgrep.android.CodeAnalysis method*), 70
code_cookies() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 135
code_cookies() (*glorifiedgrep.GlorifiedAndroid method*), 12
code_create_new_file() (*glorifiedgrep.android.CodeAnalysis method*), 70
code_create_new_file() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 135
code_create_new_file() (*glorifiedgrep.GlorifiedAndroid method*), 12
code_create_sockets() (*glorifiedgrep.android.CodeAnalysis method*), 70
code_create_sockets() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 135
code_create_sockets() (*glorifiedgrep.GlorifiedAndroid method*), 12
code_create_tempfile() (*glorified-*

grep.android.CodeAnalysis method), 71
code_create_tempfile() (glorified-grep.android.modules.malware.MalwareBehaviour method), 136
code_create_tempfile() (glorified-grep.GlorifiedAndroid method), 12
code_database_interaction() (glorified-grep.android.CodeAnalysis method), 71
code_database_interaction() (glorified-grep.android.modules.malware.MalwareBehaviour method), 136
code_database_interaction() (glorified-grep.GlorifiedAndroid method), 13
code_database_query() (glorified-grep.android.CodeAnalysis method), 71
code_database_query() (glorified-grep.android.modules.malware.MalwareBehaviour method), 136
code_database_query() (glorified-grep.GlorifiedAndroid method), 13
code_debuggable_check() (glorified-grep.android.CodeAnalysis method), 71
code_debuggable_check() (glorified-grep.android.modules.malware.MalwareBehaviour method), 136
code_debuggable_check() (glorified-grep.GlorifiedAndroid method), 13
code_debugger_check() (glorified-grep.android.CodeAnalysis method), 72
code_debugger_check() (glorified-grep.android.modules.malware.MalwareBehaviour method), 137
code_debugger_check() (glorified-grep.GlorifiedAndroid method), 13
code_deserialization() (glorified-grep.android.CodeAnalysis method), 72
code_deserialization() (glorified-grep.android.modules.malware.MalwareBehaviour method), 137
code_deserialization() (glorified-grep.GlorifiedAndroid method), 14
code_device_id() (glorified-grep.android.CodeAnalysis method), 72
code_device_id() (glorified-grep.android.modules.malware.MalwareBehaviour method), 137
code_device_id() (glorifiedgrep.GlorifiedAndroid method), 14
code_device_serial_number() (glorified-grep.android.CodeAnalysis method), 73
code_device_serial_number() (glorified-grep.android.modules.malware.MalwareBehaviour method), 138
code_device_serial_number() (glorified-grep.GlorifiedAndroid method), 14
code_download_manager() (glorified-grep.android.CodeAnalysis method), 73
code_download_manager() (glorified-grep.android.modules.malware.MalwareBehaviour method), 138
code_download_manager() (glorified-grep.GlorifiedAndroid method), 15
code_dynamic_dexclassloader() (glorified-grep.android.CodeAnalysis method), 73
code_dynamic_dexclassloader() (glorified-grep.android.modules.malware.MalwareBehaviour method), 138
code_dynamic_dexclassloader() (glorified-grep.GlorifiedAndroid method), 15
code_dynamic_other_classloader() (glorifiedgrep.android.CodeAnalysis method), 73
code_dynamic_other_classloader() (glorifiedgrep.android.modules.malware.MalwareBehaviour method), 138
code_dynamic_other_classloader() (glorifiedgrep.GlorifiedAndroid method), 15
code_exif_data() (glorified-grep.android.CodeAnalysis method), 74
code_exif_data() (glorified-grep.android.modules.malware.MalwareBehaviour method), 139
code_exif_data() (glorifiedgrep.GlorifiedAndroid method), 15
code_external_file_access() (glorified-grep.android.CodeAnalysis method), 74
code_external_file_access() (glorified-grep.android.modules.malware.MalwareBehaviour method), 139
code_external_file_access() (glorified-grep.GlorifiedAndroid method), 16
code_file_observer() (glorified-grep.android.CodeAnalysis method), 74
code_file_observer() (glorified-grep.android.modules.malware.MalwareBehaviour method), 139
code_file_observer() (glorified-grep.GlorifiedAndroid method), 16
code_file_read() (glorified-grep.android.CodeAnalysis method), 75
code_file_read() (glorified-grep.android.modules.malware.MalwareBehaviour method), 140
code_file_read() (glorifiedgrep.GlorifiedAndroid method), 16
code_file_write() (glorified-grep.android.CodeAnalysis method), 75
code_file_write() (glorified-

grep.android.modules.malware.MalwareBehaviour method), 140
code_file_write() (*glorified-grep.GlorifiedAndroid method*), 17
code_find_intents() (*glorified-grep.android.CodeAnalysis method*), 75
code_find_intents() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 140
code_find_intents() (*glorified-grep.GlorifiedAndroid method*), 17
code_firebase_imports() (*glorified-grep.android.CodeAnalysis method*), 75
code_firebase_imports() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 140
code_firebase_imports() (*glorified-grep.GlorifiedAndroid method*), 17
code_get_environment_var() (*glorified-grep.android.CodeAnalysis method*), 76
code_get_environment_var() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 141
code_get_environment_var() (*glorified-grep.GlorifiedAndroid method*), 17
code_google_api_keys() (*glorified-grep.android.CodeAnalysis method*), 76
code_google_api_keys() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 141
code_google_api_keys() (*glorified-grep.GlorifiedAndroid method*), 18
code_gps_location() (*glorified-grep.android.CodeAnalysis method*), 76
code_gps_location() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 141
code_gps_location() (*glorified-grep.GlorifiedAndroid method*), 18
code_hashing_algorithms() (*glorified-grep.android.CodeAnalysis method*), 76
code_hashing_algorithms() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 141
code_hashing_algorithms() (*glorified-grep.GlorifiedAndroid method*), 18
code_hashing_custom() (*glorified-grep.android.CodeAnalysis method*), 77
code_hashing_custom() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 142
code_hashing_custom() (*glorified-grep.GlorifiedAndroid method*), 18
code_http_request_methods() (*glorified-grep.android.CodeAnalysis method*), 77
code_http_request_methods() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 142
code_http_request_methods() (*glorified-grep.GlorifiedAndroid method*), 19
code_imports() (*glorified-grep.android.CodeAnalysis method*), 77
code_imports() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 142
code_imports() (*glorifiedgrep.GlorifiedAndroid method*), 19
code_intent_filters() (*glorified-grep.android.CodeAnalysis method*), 78
code_intent_filters() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 143
code_intent_filters() (*glorified-grep.GlorifiedAndroid method*), 19
code_intent_parameters() (*glorified-grep.android.CodeAnalysis method*), 78
code_intent_parameters() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 143
code_intent_parameters() (*glorified-grep.GlorifiedAndroid method*), 20
code_invisible_elements() (*glorified-grep.android.CodeAnalysis method*), 78
code_invisible_elements() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 143
code_invisible_elements() (*glorified-grep.GlorifiedAndroid method*), 20
code_jar_urlconnection() (*glorified-grep.android.CodeAnalysis method*), 78
code_jar_urlconnection() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 143
code_jar_urlconnection() (*glorified-grep.GlorifiedAndroid method*), 20
code_js_read_file() (*glorified-grep.android.CodeAnalysis method*), 79
code_js_read_file() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 144
code_js_read_file() (*glorified-grep.GlorifiedAndroid method*), 20
code_key_generator() (*glorified-grep.android.CodeAnalysis method*), 79
code_key_generator() (*glorified-grep.android.modules.malware.MalwareBehaviour method*), 144
code_key_generator() (*glorified-*

grep.GlorifiedAndroid method), 21
code_keystore_files() (glorified-grep.android.CodeAnalysis method), 79
code_keystore_files() (glorified-grep.android.modules.malware.MalwareBehaviour method), 144
code_keystore_files() (glorified-grep.GlorifiedAndroid method), 21
code_load_native_library() (glorified-grep.android.CodeAnalysis method), 79
code_load_native_library() (glorified-grep.android.modules.malware.MalwareBehaviour method), 144
code_load_native_library() (glorified-grep.GlorifiedAndroid method), 21
code_location() (glorified-grep.android.CodeAnalysis method), 80
code_location() (glorified-grep.android.modules.malware.MalwareBehaviour method), 145
code_location() (glorifiedgrep.GlorifiedAndroid method), 21
code_location_manager() (glorified-grep.android.CodeAnalysis method), 80
code_location_manager() (glorified-grep.android.modules.malware.MalwareBehaviour method), 145
code_location_manager() (glorified-grep.GlorifiedAndroid method), 22
code_logging() (glorified-grep.android.CodeAnalysis method), 80
code_logging() (glorified-grep.android.modules.malware.MalwareBehaviour method), 145
code_logging() (glorifiedgrep.GlorifiedAndroid method), 22
code_make_http_request() (glorified-grep.android.CodeAnalysis method), 81
code_make_http_request() (glorified-grep.android.modules.malware.MalwareBehaviour method), 146
code_make_http_request() (glorified-grep.GlorifiedAndroid method), 22
code_make_https_request() (glorified-grep.android.CodeAnalysis method), 81
code_make_https_request() (glorified-grep.android.modules.malware.MalwareBehaviour method), 146
code_make_https_request() (glorified-grep.GlorifiedAndroid method), 23
code_mediastore() (glorified-grep.android.CodeAnalysis method), 81
code_mediastore() (glorified-grep.android.modules.malware.MalwareBehaviour method), 146
code_mediastore() (glorified-grep.GlorifiedAndroid method), 23
code_notification_access() (glorified-grep.android.CodeAnalysis method), 81
code_notification_access() (glorified-grep.android.modules.malware.MalwareBehaviour method), 146
code_notification_access() (glorified-grep.GlorifiedAndroid method), 23
code_notification_manager() (glorified-grep.android.CodeAnalysis method), 82
code_notification_manager() (glorified-grep.android.modules.malware.MalwareBehaviour method), 147
code_notification_manager() (glorified-grep.GlorifiedAndroid method), 23
code_null_cipher() (glorified-grep.android.CodeAnalysis method), 82
code_null_cipher() (glorified-grep.android.modules.malware.MalwareBehaviour method), 147
code_null_cipher() (glorified-grep.GlorifiedAndroid method), 24
code_object_deserialization() (glorified-grep.android.CodeAnalysis method), 82
code_object_deserialization() (glorified-grep.android.modules.malware.MalwareBehaviour method), 147
code_object_deserialization() (glorified-grep.GlorifiedAndroid method), 24
code_package_installed() (glorified-grep.android.CodeAnalysis method), 82
code_package_installed() (glorified-grep.android.modules.malware.MalwareBehaviour method), 147
code_package_installed() (glorified-grep.GlorifiedAndroid method), 24
code_parse_uri() (glorified-grep.android.CodeAnalysis method), 83
code_parse_uri() (glorified-grep.android.modules.malware.MalwareBehaviour method), 148
code_parse_uri() (glorifiedgrep.GlorifiedAndroid method), 24
code_password_finder() (glorified-grep.android.CodeAnalysis method), 83
code_password_finder() (glorified-grep.android.modules.malware.MalwareBehaviour method), 148
code_password_finder() (glorified-grep.GlorifiedAndroid method), 25
code_phone_sensors() (glorified-grep.android.CodeAnalysis method), 83

code_phone_sensors() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 148
code_phone_sensors() (*glorifiedgrep.GlorifiedAndroid method*), 25
code_rabbit_amqp() (*glorifiedgrep.android.CodeAnalysis method*), 84
code_rabbit_amqp() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 149
code_rabbit_amqp() (*glorifiedgrep.GlorifiedAndroid method*), 25
code_read_sms_messages() (*glorifiedgrep.android.CodeAnalysis method*), 84
code_read_sms_messages() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 149
code_read_sms_messages() (*glorifiedgrep.GlorifiedAndroid method*), 26
code_reflection() (*glorifiedgrep.android.CodeAnalysis method*), 84
code_reflection() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 149
code_reflection() (*glorifiedgrep.GlorifiedAndroid method*), 26
code_regex_matcher() (*glorifiedgrep.android.CodeAnalysis method*), 84
code_regex_matcher() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 149
code_regex_matcher() (*glorifiedgrep.GlorifiedAndroid method*), 26
code_regex_pattern() (*glorifiedgrep.android.CodeAnalysis method*), 85
code_regex_pattern() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 150
code_regex_pattern() (*glorifiedgrep.GlorifiedAndroid method*), 26
code_root_access() (*glorifiedgrep.android.CodeAnalysis method*), 85
code_root_access() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 150
code_root_access() (*glorifiedgrep.GlorifiedAndroid method*), 27
code_screenshots() (*glorifiedgrep.android.CodeAnalysis method*), 85
code_screenshots() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 150
code_screenshots() (*glorifiedgrep.GlorifiedAndroid method*), 27
code_sdcard() (*glorifiedgrep.android.CodeAnalysis method*), 85
code_sdcard() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 150
code_sdcard() (*glorifiedgrep.GlorifiedAndroid method*), 27
code_search() (*glorifiedgrep.android.CodeAnalysis method*), 86
code_search() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 151
code_search() (*glorifiedgrep.GlorifiedAndroid method*), 27
code_send_sms_text() (*glorifiedgrep.android.CodeAnalysis method*), 86
code_send_sms_text() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 151
code_send_sms_text() (*glorifiedgrep.GlorifiedAndroid method*), 28
code_services() (*glorifiedgrep.android.CodeAnalysis method*), 86
code_services() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 151
code_services() (*glorifiedgrep.GlorifiedAndroid method*), 28
code_shared_preferences() (*glorifiedgrep.android.CodeAnalysis method*), 87
code_shared_preferences() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 152
code_shared_preferences() (*glorifiedgrep.GlorifiedAndroid method*), 28
code_sim_information() (*glorifiedgrep.android.CodeAnalysis method*), 87
code_sim_information() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 152
code_sim_information() (*glorifiedgrep.GlorifiedAndroid method*), 29
code_sql_injection_points() (*glorifiedgrep.android.CodeAnalysis method*), 87
code_sql_injection_points() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 152
code_sql_injection_points() (*glorifiedgrep.GlorifiedAndroid method*), 29
code_sql_injection_user_input() (*glorifiedgrep.android.CodeAnalysis method*), 87
code_sql_injection_user_input() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 152

code_sql_injection_user_input() (*glorifiedgrep.GlorifiedAndroid method*), 29
code_sql_java_implementation() (*glorifiedgrep.android.CodeAnalysis method*), 88
code_sql_java_implementation() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 153
code_sql_java_implementation() (*glorifiedgrep.GlorifiedAndroid method*), 29
code_sql_query_other() (*glorifiedgrep.android.CodeAnalysis method*), 88
code_sql_query_other() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 153
code_sql_query_other() (*glorifiedgrep.GlorifiedAndroid method*), 30
code_sql_select_raw_query() (*glorifiedgrep.android.CodeAnalysis method*), 88
code_sql_select_raw_query() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 153
code_sql_select_raw_query() (*glorifiedgrep.GlorifiedAndroid method*), 30
code_sqlcipher_password() (*glorifiedgrep.android.CodeAnalysis method*), 89
code_sqlcipher_password() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 154
code_sqlcipher_password() (*glorifiedgrep.GlorifiedAndroid method*), 30
code_sqlite_operations() (*glorifiedgrep.android.CodeAnalysis method*), 89
code_sqlite_operations() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 154
code_sqlite_operations() (*glorifiedgrep.GlorifiedAndroid method*), 31
code_ssl_connections() (*glorifiedgrep.android.CodeAnalysis method*), 89
code_ssl_connections() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 154
code_ssl_connections() (*glorifiedgrep.GlorifiedAndroid method*), 31
code_stack_trace() (*glorifiedgrep.android.CodeAnalysis method*), 89
code_stack_trace() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 154
code_stack_trace() (*glorifiedgrep.GlorifiedAndroid method*), 31
code_static_iv() (*glorifiedgrep.android.CodeAnalysis method*), 90
code_static_iv() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 155
code_static_iv() (*glorifiedgrep.GlorifiedAndroid method*), 31
code_string_constants() (*glorifiedgrep.android.CodeAnalysis method*), 90
code_string_constants() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 155
code_string_constants() (*glorifiedgrep.GlorifiedAndroid method*), 32
code_stub_packed() (*glorifiedgrep.android.CodeAnalysis method*), 90
code_stub_packed() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 155
code_stub_packed() (*glorifiedgrep.GlorifiedAndroid method*), 32
code_system_file_exists() (*glorifiedgrep.android.CodeAnalysis method*), 90
code_system_file_exists() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 155
code_system_file_exists() (*glorifiedgrep.GlorifiedAndroid method*), 32
code_system_service() (*glorifiedgrep.android.CodeAnalysis method*), 91
code_system_service() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 156
code_system_service() (*glorifiedgrep.GlorifiedAndroid method*), 32
code_tcp_sockets() (*glorifiedgrep.android.CodeAnalysis method*), 91
code_tcp_sockets() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 156
code_tcp_sockets() (*glorifiedgrep.GlorifiedAndroid method*), 33
code_trust_all_ssl() (*glorifiedgrep.android.CodeAnalysis method*), 91
code_trust_all_ssl() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 156
code_trust_all_ssl() (*glorifiedgrep.GlorifiedAndroid method*), 33
code_udp_sockets() (*glorifiedgrep.android.CodeAnalysis method*), 92
code_udp_sockets() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 157
code_udp_sockets() (*glorifiedgrep.GlorifiedAndroid method*), 33
code_weak_hashing() (*glorified-*

grep.android.CodeAnalysis method), 92
code_weak_hashing() (glorified-grep.android.modules.malware.MalwareBehaviour method), 157
code_weak_hashing() (glorified-grep.GlorifiedAndroid method), 33
code_websocket_usage() (glorified-grep.android.CodeAnalysis method), 92
code_websocket_usage() (glorified-grep.android.modules.malware.MalwareBehaviour method), 157
code_websocket_usage() (glorified-grep.GlorifiedAndroid method), 34
code_webview_content_access() (glorified-grep.android.CodeAnalysis method), 92
code_webview_content_access() (glorified-grep.android.modules.malware.MalwareBehaviour method), 157
code_webview_content_access() (glorified-grep.GlorifiedAndroid method), 34
code_webview_database() (glorified-grep.android.CodeAnalysis method), 93
code_webview_database() (glorified-grep.android.modules.malware.MalwareBehaviour method), 158
code_webview_database() (glorified-grep.GlorifiedAndroid method), 34
code_webview_debug_enabled() (glorified-grep.android.CodeAnalysis method), 93
code_webview_debug_enabled() (glorified-grep.android.modules.malware.MalwareBehaviour method), 158
code_webview_debug_enabled() (glorified-grep.GlorifiedAndroid method), 35
code_webview_file_access() (glorified-grep.android.CodeAnalysis method), 93
code_webview_file_access() (glorified-grep.android.modules.malware.MalwareBehaviour method), 158
code_webview_file_access() (glorified-grep.GlorifiedAndroid method), 35
code_webview_get_request() (glorified-grep.android.CodeAnalysis method), 93
code_webview_get_request() (glorified-grep.android.modules.malware.MalwareBehaviour method), 158
code_webview_get_request() (glorified-grep.GlorifiedAndroid method), 35
code_webview_js_enabled() (glorified-grep.android.CodeAnalysis method), 94
code_webview_js_enabled() (glorified-grep.android.modules.malware.MalwareBehaviour method), 159
code_webview_js_enabled() (glorified-grep.GlorifiedAndroid method), 35
code_webview_post_request() (glorified-grep.android.CodeAnalysis method), 94
code_webview_post_request() (glorified-grep.android.modules.malware.MalwareBehaviour method), 159
code_webview_post_request() (glorified-grep.GlorifiedAndroid method), 36
code_xml_processor() (glorified-grep.android.CodeAnalysis method), 94
code_xml_processor() (glorified-grep.android.modules.malware.MalwareBehaviour method), 159
code_xml_processor() (glorified-grep.GlorifiedAndroid method), 36
code_xor_encryption() (glorified-grep.android.CodeAnalysis method), 95
code_xor_encryption() (glorified-grep.android.modules.malware.MalwareBehaviour method), 160
code_xor_encryption() (glorified-grep.GlorifiedAndroid method), 36
code_xpath() (glorifiedgrep.android.CodeAnalysis method), 95
code_xpath() (glorified-grep.android.modules.malware.MalwareBehaviour method), 160
code_xpath() (glorifiedgrep.GlorifiedAndroid method), 36
CodeAnalysis (class in glorifiedgrep.android), 63
count (glorifiedgrep.out.GreppedOut attribute), 173

D

dex_classes() (glorified-grep.android.modules.utils.NativeDEXAnalysis method), 168
dex_info() (glorified-grep.android.modules.utils.NativeDEXAnalysis method), 169
dex_methods() (glorified-grep.android.modules.utils.NativeDEXAnalysis method), 169
dex_parse() (glorified-grep.android.modules.utils.NativeDEXAnalysis method), 169
dex_strings() (glorified-grep.android.modules.utils.NativeDEXAnalysis method), 169

E

elf_exported_symbols() (glorified-grep.android.modules.utils.NativeELFAnalysis method), 167

elf_header_info() (*glorifiedgrep.android.modules.utils.NativeELFAnalysis method*), 167
elf_imported_symbols() (*glorifiedgrep.android.modules.utils.NativeELFAnalysis method*), 167
elf_libraries_binary() (*glorifiedgrep.android.modules.utils.NativeELFAnalysis method*), 168
elf_strings_from_binary() (*glorifiedgrep.android.modules.utils.NativeELFAnalysis method*), 168
exclude_file() (*glorifiedgrep.out.GreppedOut method*), 173
exodus_trackers() (*glorifiedgrep.android.OtherAnalysis class method*), 121
exodus_trackers() (*glorifiedgrep.GlorifiedAndroid class method*), 37

F

file_activities_handling_passwords() (*glorifiedgrep.GlorifiedAndroid method*), 37
file_database_file_paths() (*glorifiedgrep.GlorifiedAndroid method*), 37
file_get_file_types() (*glorifiedgrep.GlorifiedAndroid method*), 37
file_get_java_classes() (*glorifiedgrep.GlorifiedAndroid method*), 38
file_hash_of_apk() (*glorifiedgrep.GlorifiedAndroid method*), 38
file_html_files() (*glorifiedgrep.GlorifiedAndroid method*), 38
file_interesting() (*glorifiedgrep.GlorifiedAndroid method*), 38
file_jar_files() (*glorifiedgrep.GlorifiedAndroid method*), 39
file_js_files() (*glorifiedgrep.GlorifiedAndroid method*), 39
file_kivy_app() (*glorifiedgrep.GlorifiedAndroid method*), 39
file_native_code() (*glorifiedgrep.GlorifiedAndroid method*), 39
file_other_langs() (*glorifiedgrep.GlorifiedAndroid method*), 40
file_react_app() (*glorifiedgrep.GlorifiedAndroid method*), 40
file_res_strings() (*glorifiedgrep.GlorifiedAndroid method*), 40
file_resource_xml() (*glorifiedgrep.GlorifiedAndroid method*), 40
file_shared_libs_file_paths() (*glorifiedgrep.GlorifiedAndroid method*), 40
file_xml_files() (*glorifiedgrep.GlorifiedAndroid method*), 41
files (*glorifiedgrep.out.GreppedOut attribute*), 173

G

GlorifiedAndroid (*class in glorifiedgrep*), 1
GreppedOut (*class in glorifiedgrep.out*), 173

I

in_file() (*glorifiedgrep.out.GreppedOut method*), 173

J

JKS (*class in glorifiedgrep.android.modules.utils*), 165
jks_certificate() (*glorifiedgrep.android.modules.utils.JKS method*), 165
jks_key_alias() (*glorifiedgrep.android.modules.utils.JKS method*), 165
jks_password_bruteforce() (*glorifiedgrep.android.modules.utils.Utils method*), 171
jks_private_key() (*glorifiedgrep.android.modules.utils.JKS method*), 166

M

malware_access_call_logs() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 160
malware_access_camera() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 160
malware_accessibility_services() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 161
malware_boot_completed_persistance() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 161
malware_browser_db_access() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 161
malware_database_query() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 161
malware_debug() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 162
malware_download_files() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 162

malware_get_external_storage() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 162
malware_get_installed_packages() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 162
malware_obtain_file_metadata() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 163
malware_screen_unlock() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 163
malware_take_screenshots() (*glorifiedgrep.android.modules.malware.MalwareBehaviour method*), 163
MalwareBehaviour (*class in glorifiedgrep.android.modules.malware*), 127
manifest_activities() (*glorifiedgrep.android.ParseManifest method*), 103
manifest_activities() (*glorifiedgrep.GlorifiedAndroid method*), 41
manifest_activity_alias() (*glorifiedgrep.android.ParseManifest method*), 103
manifest_activity_alias() (*glorifiedgrep.GlorifiedAndroid method*), 41
manifest_allow_backup() (*glorifiedgrep.android.ParseManifest method*), 104
manifest_allow_backup() (*glorifiedgrep.GlorifiedAndroid method*), 41
manifest_android_version() (*glorifiedgrep.android.ParseManifest method*), 104
manifest_android_version() (*glorifiedgrep.GlorifiedAndroid method*), 42
manifest_application_node() (*glorifiedgrep.android.ParseManifest method*), 104
manifest_application_node() (*glorifiedgrep.GlorifiedAndroid method*), 42
manifest_bind_permissions() (*glorifiedgrep.android.ParseManifest method*), 104
manifest_bind_permissions() (*glorifiedgrep.GlorifiedAndroid method*), 42
manifest_custom_permission() (*glorifiedgrep.android.ParseManifest method*), 105
manifest_custom_permission() (*glorifiedgrep.GlorifiedAndroid method*), 42
manifest_dangerous_permission() (*glorifiedgrep.android.ParseManifest method*), 105
manifest_dangerous_permission() (*glorifiedgrep.GlorifiedAndroid method*), 43
manifest_debuggable() (*glorifiedgrep.android.ParseManifest method*), 105
manifest_debuggable() (*glorifiedgrep.GlorifiedAndroid method*), 43
manifest_exported_providers() (*glorifiedgrep.android.ParseManifest method*), 105
manifest_exported_providers() (*glorifiedgrep.GlorifiedAndroid method*), 43
manifest_intent_uri_filter() (*glorifiedgrep.android.ParseManifest method*), 106
manifest_intent_uri_filter() (*glorifiedgrep.GlorifiedAndroid method*), 43
manifest_main_activity() (*glorifiedgrep.android.ParseManifest method*), 106
manifest_main_activity() (*glorifiedgrep.GlorifiedAndroid method*), 44
manifest_meta_data() (*glorifiedgrep.android.ParseManifest method*), 106
manifest_meta_data() (*glorifiedgrep.GlorifiedAndroid method*), 44
manifest_min_sdk() (*glorifiedgrep.android.ParseManifest method*), 106
manifest_min_sdk() (*glorifiedgrep.GlorifiedAndroid method*), 44
manifest_package_name() (*glorifiedgrep.android.ParseManifest method*), 107
manifest_package_name() (*glorifiedgrep.GlorifiedAndroid method*), 44
manifest_permission() (*glorifiedgrep.android.ParseManifest method*), 107
manifest_permission() (*glorifiedgrep.GlorifiedAndroid method*), 45
manifest_platform_build_version_code() (*glorifiedgrep.android.ParseManifest method*), 107
manifest_platform_build_version_code() (*glorifiedgrep.GlorifiedAndroid method*), 45
manifest_platform_build_version_name() (*glorifiedgrep.android.ParseManifest method*), 107
manifest_platform_build_version_name() (*glorifiedgrep.GlorifiedAndroid method*), 45
manifest_providers() (*glorifiedgrep.android.ParseManifest method*), 108
manifest_providers() (*glorifiedgrep.GlorifiedAndroid method*), 45
manifest_receivers() (*glorifiedgrep.android.ParseManifest method*), 108
manifest_receivers() (*glorifiedgrep.GlorifiedAndroid method*), 46
manifest_secrets() (*glorifiedgrep.android.ParseManifest method*), 108
manifest_secrets() (*glorifiedgrep.GlorifiedAndroid method*), 46
manifest_services() (*glorifiedgrep.android.ParseManifest method*), 108
manifest_services() (*glorifiedgrep.GlorifiedAndroid method*), 46
manifest_signature_permission() (*glorified-*

grep.android.ParseManifest method), 109
manifest_signature_permission() (glorified-grep.GlorifiedAndroid method), 46
manifest_target_sdk() (glorified-grep.android.ParseManifest method), 109
manifest_target_sdk() (glorified-grep.GlorifiedAndroid method), 47
manifest_uses_configuration() (glorified-grep.android.ParseManifest method), 109
manifest_uses_configuration() (glorified-grep.GlorifiedAndroid method), 47
manifest_uses_feature() (glorified-grep.android.ParseManifest method), 109
manifest_uses_feature() (glorified-grep.GlorifiedAndroid method), 47
manifest_uses_library() (glorified-grep.android.ParseManifest method), 110
manifest_uses_library() (glorified-grep.GlorifiedAndroid method), 47
manifest_uses_permission() (glorified-grep.android.ParseManifest method), 110
manifest_uses_permission() (glorified-grep.GlorifiedAndroid method), 48
manifest_version_code() (glorified-grep.android.ParseManifest method), 110
manifest_version_code() (glorified-grep.GlorifiedAndroid method), 48
manifest_version_name() (glorified-grep.android.ParseManifest method), 110
manifest_version_name() (glorified-grep.GlorifiedAndroid method), 48
matches (glorifiedgrep.out.GreppedOut attribute), 174

N

NativeDEXAnalysis (class in glorified-grep.android.modules.utils), 168
NativeELFAnalysis (class in glorified-grep.android.modules.utils), 167

O

other_ad_networks() (glorified-grep.android.OtherAnalysis method), 122
other_ad_networks() (glorified-grep.GlorifiedAndroid method), 48
other_all_urls() (glorified-grep.android.OtherAnalysis method), 122
other_all_urls() (glorifiedgrep.GlorifiedAndroid method), 49
other_aws_keys() (glorified-grep.android.OtherAnalysis method), 122
other_aws_keys() (glorifiedgrep.GlorifiedAndroid method), 49
other_content_urlhandler() (glorified-grep.android.OtherAnalysis method), 122
other_content_urlhandler() (glorified-grep.GlorifiedAndroid method), 49
other_email_addresses() (glorified-grep.android.OtherAnalysis method), 123
other_email_addresses() (glorified-grep.GlorifiedAndroid method), 49
other_file_urlhandler() (glorified-grep.android.OtherAnalysis method), 123
other_file_urlhandler() (glorified-grep.GlorifiedAndroid method), 50
other_find_trackers_ads() (glorified-grep.android.OtherAnalysis method), 123
other_find_trackers_ads() (glorified-grep.GlorifiedAndroid method), 50
other_github_token() (glorified-grep.android.OtherAnalysis method), 123
other_github_token() (glorified-grep.GlorifiedAndroid method), 50
other_google_ads_import() (glorified-grep.android.OtherAnalysis method), 124
other_google_ads_import() (glorified-grep.GlorifiedAndroid method), 50
other_http_urls() (glorified-grep.android.OtherAnalysis method), 124
other_http_urls() (glorified-grep.GlorifiedAndroid method), 51
other_ip_address() (glorified-grep.android.OtherAnalysis method), 124
other_ip_address() (glorified-grep.GlorifiedAndroid method), 51
other_password_in_url() (glorified-grep.android.OtherAnalysis method), 124
other_password_in_url() (glorified-grep.GlorifiedAndroid method), 51
other_secret_keys() (glorified-grep.android.OtherAnalysis method), 125
other_secret_keys() (glorified-grep.GlorifiedAndroid method), 51
other_unicode_chars() (glorified-grep.android.OtherAnalysis method), 125
other_unicode_chars() (glorified-grep.GlorifiedAndroid method), 52
other_websocket_urlhandler() (glorified-grep.android.OtherAnalysis method), 125
other_websocket_urlhandler() (glorified-grep.GlorifiedAndroid method), 52
OtherAnalysis (class in glorifiedgrep.android), 121
owasp_cloud_backup() (glorified-grep.android.CodeAnalysis method), 95
owasp_cloud_backup() (glorified-grep.android.OWASPAnalysis method), 113
owasp_cloud_backup() (glorified-grep.GlorifiedAndroid method), 52
owasp_code_check_permission() (glorified-

grep.android.CodeAnalysis method), 95
owasp_code_check_permission() (*glorifiedgrep.android.OWASPAnalysis method*), 114
owasp_code_check_permission() (*glorifiedgrep.GlorifiedAndroid method*), 53
owasp_crypto_imports() (*glorifiedgrep.android.CodeAnalysis method*), 96
owasp_crypto_imports() (*glorifiedgrep.android.OWASPAnalysis method*), 114
owasp_crypto_imports() (*glorifiedgrep.GlorifiedAndroid method*), 53
owasp_crypto_primitives() (*glorifiedgrep.android.CodeAnalysis method*), 96
owasp_crypto_primitives() (*glorifiedgrep.android.OWASPAnalysis method*), 114
owasp_crypto_primitives() (*glorifiedgrep.GlorifiedAndroid method*), 53
owasp_debug_code() (*glorifiedgrep.android.CodeAnalysis method*), 96
owasp_debug_code() (*glorifiedgrep.android.OWASPAnalysis method*), 115
owasp_debug_code() (*glorifiedgrep.GlorifiedAndroid method*), 54
owasp_encrypted_sql_db() (*glorifiedgrep.android.CodeAnalysis method*), 97
owasp_encrypted_sql_db() (*glorifiedgrep.android.OWASPAnalysis method*), 115
owasp_encrypted_sql_db() (*glorifiedgrep.GlorifiedAndroid method*), 54
owasp_external_cache_dir() (*glorifiedgrep.android.CodeAnalysis method*), 97
owasp_external_cache_dir() (*glorifiedgrep.android.OWASPAnalysis method*), 115
owasp_external_cache_dir() (*glorifiedgrep.GlorifiedAndroid method*), 54
owasp_external_storage() (*glorifiedgrep.android.CodeAnalysis method*), 97
owasp_external_storage() (*glorifiedgrep.android.OWASPAnalysis method*), 115
owasp_external_storage() (*glorifiedgrep.GlorifiedAndroid method*), 54
owasp_get_secret_keys() (*glorifiedgrep.android.CodeAnalysis method*), 97
owasp_get_secret_keys() (*glorifiedgrep.android.OWASPAnalysis method*), 116
owasp_get_secret_keys() (*glorifiedgrep.GlorifiedAndroid method*), 55
owasp_hardcoded_keys() (*glorifiedgrep.android.CodeAnalysis method*), 98
owasp_hardcoded_keys() (*glorifiedgrep.android.OWASPAnalysis method*), 116
owasp_hardcoded_keys() (*glorifiedgrep.GlorifiedAndroid method*), 55
owasp_insecure_fingerprint_auth() (*glorifiedgrep.android.CodeAnalysis method*), 98
owasp_insecure_fingerprint_auth() (*glorifiedgrep.android.OWASPAnalysis method*), 116
owasp_insecure_fingerprint_auth() (*glorifiedgrep.GlorifiedAndroid method*), 55
owasp_insecure_random() (*glorifiedgrep.android.CodeAnalysis method*), 98
owasp_insecure_random() (*glorifiedgrep.android.OWASPAnalysis method*), 117
owasp_insecure_random() (*glorifiedgrep.GlorifiedAndroid method*), 56
owasp_intent_parameter() (*glorifiedgrep.android.CodeAnalysis method*), 99
owasp_intent_parameter() (*glorifiedgrep.android.OWASPAnalysis method*), 117
owasp_intent_parameter() (*glorifiedgrep.GlorifiedAndroid method*), 56
owasp_keychain_password() (*glorifiedgrep.android.CodeAnalysis method*), 99
owasp_keychain_password() (*glorifiedgrep.android.OWASPAnalysis method*), 117
owasp_keychain_password() (*glorifiedgrep.GlorifiedAndroid method*), 56
owasp_keystore_cert_pinning() (*glorifiedgrep.android.CodeAnalysis method*), 99
owasp_keystore_cert_pinning() (*glorifiedgrep.android.OWASPAnalysis method*), 117
owasp_keystore_cert_pinning() (*glorifiedgrep.GlorifiedAndroid method*), 56
owasp_properly_signed() (*glorifiedgrep.android.CodeAnalysis method*), 99
owasp_properly_signed() (*glorifiedgrep.android.OWASPAnalysis method*), 118
owasp_properly_signed() (*glorifiedgrep.GlorifiedAndroid method*), 57
owasp_runtime_exception_handling() (*glorifiedgrep.android.CodeAnalysis method*), 100
owasp_runtime_exception_handling() (*glorifiedgrep.android.OWASPAnalysis method*), 118
owasp_runtime_exception_handling() (*glorifiedgrep.GlorifiedAndroid method*), 57
owasp_ssl_no_hostname_verification() (*glorifiedgrep.android.CodeAnalysis method*), 100
owasp_ssl_no_hostname_verification() (*glorifiedgrep.android.OWASPAnalysis method*), 118
owasp_ssl_no_hostname_verification() (*glorifiedgrep.GlorifiedAndroid method*), 57
owasp_webview_cert_pinning() (*glorifiedgrep.android.CodeAnalysis method*), 100
owasp_webview_cert_pinning() (*glorifiedgrep.android.OWASPAnalysis method*), 119

owasp_webview_cert_pinning() (*glorifiedgrep.GlorifiedAndroid method*), 58
owasp_webview_loadurl() (*glorifiedgrep.android.CodeAnalysis method*), 101
owasp_webview_loadurl() (*glorifiedgrep.android.OWASPAnalysis method*), 119
owasp_webview_loadurl() (*glorifiedgrep.GlorifiedAndroid method*), 58
owasp_webview_native_function() (*glorifiedgrep.android.CodeAnalysis method*), 101
owasp_webview_native_function() (*glorifiedgrep.android.OWASPAnalysis method*), 119
owasp_webview_native_function() (*glorifiedgrep.GlorifiedAndroid method*), 58
owasp_webview_ssl_ignore() (*glorifiedgrep.android.CodeAnalysis method*), 101
owasp_webview_ssl_ignore() (*glorifiedgrep.android.OWASPAnalysis method*), 119
owasp_webview_ssl_ignore() (*glorifiedgrep.GlorifiedAndroid method*), 58
owasp_world_read_write_files() (*glorifiedgrep.android.CodeAnalysis method*), 101
owasp_world_read_write_files() (*glorifiedgrep.android.OWASPAnalysis method*), 120
owasp_world_read_write_files() (*glorifiedgrep.GlorifiedAndroid method*), 59
OWASPAnalysis (*class in glorifiedgrep.android*), 113

P

ParseManifest (*class in glorifiedgrep.android*), 103

S

search_methods() (*glorifiedgrep.GlorifiedAndroid method*), 59
SQL (*class in glorifiedgrep.android.modules.utils*), 170
sqldb_dump_database() (*glorifiedgrep.android.modules.utils.SQL method*), 170
sqldb_table_column_names() (*glorifiedgrep.android.modules.utils.SQL method*), 170
sqldb_table_data() (*glorifiedgrep.android.modules.utils.SQL method*), 170
sqldb_tables() (*glorifiedgrep.android.modules.utils.SQL method*), 170

U

Utils (*class in glorifiedgrep.android.modules.utils*), 171
utils_xml_to_dict() (*glorifiedgrep.android.modules.utils.Utils method*), 172