Katana

Release 1.0

Feb 26, 2022

Contents:

`katana` is a command-line utility that automates checking the "low-hanging fruit" in a CTF challenge. Written in Python, it is intended to help an individual do things they might otherwise forget to do.

A lot of the context and ideas for this tool come from the living document available at https://github.com/JohnHammond/ctf-katana

CHAPTER 1

Installation Instructions

Katana is designed first and foremost as a Python module. A `setup.py` script is provided to install via `setuptools`. There are a number of binary dependencies which individual units depend on. When running Katana, you will be notified of these dependencies if they are missing. A short list is provided below, but may not be up to date depending on the units currently installed.

1.1 Binary Dependencies

Depending on your distribution, installation methods will differ. In general, you will require the following packages:

- Python3.7+
- Python3 setuptools
- Python3 pip
- Python3 virtualenv (for development)
- libffi-dev
- libssl-dev
- pandoc
- libgmp3-dev
- libzbar-dev
- tesseract-ocr
- xsel
- libpoppler-cpp-dev
- libmpc-dev

If you are using Ubuntu, these requirements can be installed with the following `apt` command:

```
sudo apt install -y python3.7-tk tk-dev python3.7 python3-pip python3-setuptools␣
↪python3.7-dev \
    python3.7-venv libffi-dev libssl-dev pandoc libgmp3-dev libzbar-dev tesseract-ocr␣
↪xsel \
    libpoppler-cpp-dev libmpc-dev
```

Installation on other distributions may differ (e.g. `yum` for CentOS, `pacman` for Arch, etc). Also, the names of individual packages may differ. Consult your distribution package manager for locating these dependencies.

1.2 Installing Katana

To install both the Katana module and Read-Evaluate-Print-Loop (REPL) interpreter, use setup tools:

```
python setup.py install
```

This will install Katana and all of it's Python dependencies in your current environment.

1.3 External Unit Dependencies

On your first few runs of Katana, you may find that you receive dependency errors related to binaries not present on your system. These dependencies are specific to the units you have installed. The default units used by Katana have the following system dependencies. Installation of these packages varies by package and distribution. Consult your distribution documentation for further assistance in installing them.

- exiftool
- steghide
- stegsnow
- zsteg
- jsteg
- node
- binwalk
- foremost
- unzip
- npiet
- tcpflow
- git
- apktool
- tesseract
- qpdf
- pdfinfo
- pdfimages
- strings

CHAPTER 2

Getting Started

Katana can be used in a number of different ways. It was designed first as a framework which is importable into other projects, however it provides a built-in interface in the form of a REPL.

2.1 Using the REPL

The Katana REPL is available by simply running the Katana module or through the `setuptools` script:

```
# Run as a python module
python -m katana ...
# Or using the bundled setuptools script
katana ...
```

The REPL provides all the features of the Katana module plus some extras, and is implemented using the `cmd2` Python module. All commands are documented within the REPL itself, and the you can find the most up to date help by running the `help` command from within the interpreter. At the time of writing, the following runtime arguments may be supplied:

```
usage: katana [-h] [--config CONFIG] [--manager MANAGER] [--timeout TIMEOUT]
           [--auto] [--unit UNIT] [--exclude EXCLUDE] [--flag FLAG]
           [--force] [--apktool APKTOOL] [--md5 MD5] [--affine AFFINE]
           [--atbash ATBASH] [--caesar CAESAR] [--caesar255 CAESAR255]
           [--dna DNA] [--phonetic PHONETIC] [--polybius POLYBIUS]
           [--quipqiup QUIPQIUP] [--railfence RAILFENCE]
           [--reverse REVERSE] [--rot47 ROT47] [--rsa RSA] [--t9 T9]
           [--vigenere VIGENERE] [--xor XOR] [--brainfuck BRAINFUCK]
           [--cow COW] [--jsfuck JSFUCK] [--malbolge MALBOLGE] [--ook OOK]
           [--piet PIET] [--pikalang PIKALANG] [--binwalk BINWALK]
           [--foremost FOREMOST] [--gunzip GUNZIP] [--tesseract TESSERACT]
           [--tcpflow TCPFLOW] [--pdf2text PDF2TEXT] [--pdfcrack PDFCRACK]
           [--pdfimages PDFIMAGES] [--pdfinfo PDFINFO] [--ascii85 ASCII85]
           [--base32 BASE32] [--base58 BASE58] [--base64 BASE64]
           [--base85 BASE85] [--exiftool EXIFTOOL] [--morsecode MORSECODE]
```

(continues on next page)

(continued from previous page)

```
          [--qrcode QRCODE] [--strings STRINGS] [--unbinary UNBINARY]
          [--undecimal UNDECIMAL] [--unhexlify UNHEXLIFY]
          [--urldecode URLDECODE] [--audio_spectrogram AUDIO_SPECTROGRAM]
          [--dtmf_decode DTMF_DECODE] [--jsteg JSTEG] [--snow SNOW]
          [--steghide STEGHIDE] [--stegsnow STEGSNOW]
          [--stegsolve STEGSOLVE] [--whitespace WHITESPACE]
          [--zsteg ZSTEG] [--extract EXTRACT]
          [--basic_img_shell BASIC_IMG_SHELL]
          [--basic_nosqli BASIC_NOSQLI] [--basic_sqli BASIC_SQLI]
          [--cookies COOKIES] [--form_submit FORM_SUBMIT] [--git GIT]
          [--logon_cookies LOGON_COOKIES] [--robots ROBOTS]
          [--spider SPIDER] [--unzip UNZIP]
          [targets [targets ...]]

Automatically identify and solve basic Capture the Flag challenges

positional arguments:
  targets               targets to evaluate

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG, -c CONFIG
                        configuration file
  --manager MANAGER, -m MANAGER
                        comma separated manager configurations (e.g. flag-
                        format=FLAG{.*?})
  --timeout TIMEOUT, -t TIMEOUT
                        timeout for all unit evaluations in seconds
  --auto, -a            shorthand for `-m auto=True`
  --unit UNIT, -u UNIT  explicitly run a unit on target
  --exclude EXCLUDE, -e EXCLUDE
                        exclude a unit from running
  --flag FLAG, -f FLAG  set the flag format
  --force               Force execution even if results directory exists
  --apktool APKTOOL     comma separated unit configuration
  --md5 MD5             comma separated unit configuration
  --affine AFFINE       comma separated unit configuration
  --atbash ATBASH       comma separated unit configuration
  --caesar CAESAR       comma separated unit configuration
  --caesar255 CAESAR255
                        comma separated unit configuration
  --dna DNA             comma separated unit configuration
  --phonetic PHONETIC   comma separated unit configuration
  --polybius POLYBIUS   comma separated unit configuration
  --quipqiup QUIPQIUP   comma separated unit configuration
  --railfence RAILFENCE
                        comma separated unit configuration
  --reverse REVERSE     comma separated unit configuration
  --rot47 ROT47         comma separated unit configuration
  --rsa RSA             comma separated unit configuration
  --t9 T9               comma separated unit configuration
  --vigenere VIGENERE   comma separated unit configuration
  --xor XOR             comma separated unit configuration
  --brainfuck BRAINFUCK
                        comma separated unit configuration
  --cow COW             comma separated unit configuration
  --jsfuck JSFUCK       comma separated unit configuration
```

(continues on next page)

(continued from previous page)

```
  --malbolge MALBOLGE   comma separated unit configuration
  --ook OOK             comma separated unit configuration
  --piet PIET           comma separated unit configuration
  --pikalang PIKALANG   comma separated unit configuration
  --binwalk BINWALK     comma separated unit configuration
  --foremost FOREMOST   comma separated unit configuration
  --gunzip GUNZIP       comma separated unit configuration
  --tesseract TESSERACT
                        comma separated unit configuration
  --tcpflow TCPFLOW     comma separated unit configuration
  --pdf2text PDF2TEXT   comma separated unit configuration
  --pdfcrack PDFCRACK   comma separated unit configuration
  --pdfimages PDFIMAGES
                        comma separated unit configuration
  --pdfinfo PDFINFO     comma separated unit configuration
  --ascii85 ASCII85     comma separated unit configuration
  --base32 BASE32       comma separated unit configuration
  --base58 BASE58       comma separated unit configuration
  --base64 BASE64       comma separated unit configuration
  --base85 BASE85       comma separated unit configuration
  --exiftool EXIFTOOL   comma separated unit configuration
  --morsecode MORSECODE
                        comma separated unit configuration
  --qrcode QRCODE       comma separated unit configuration
  --strings STRINGS     comma separated unit configuration
  --unbinary UNBINARY   comma separated unit configuration
  --undecimal UNDECIMAL
                        comma separated unit configuration
  --unhexlify UNHEXLIFY
                        comma separated unit configuration
  --urldecode URLDECODE
                        comma separated unit configuration
  --audio_spectrogram AUDIO_SPECTROGRAM
                        comma separated unit configuration
  --dtmf_decode DTMF_DECODE
                        comma separated unit configuration
  --jsteg JSTEG         comma separated unit configuration
  --snow SNOW           comma separated unit configuration
  --steghide STEGHIDE   comma separated unit configuration
  --stegsnow STEGSNOW   comma separated unit configuration
  --stegsolve STEGSOLVE
                        comma separated unit configuration
  --whitespace WHITESPACE
                        comma separated unit configuration
  --zsteg ZSTEG         comma separated unit configuration
  --extract EXTRACT     comma separated unit configuration
  --basic_img_shell BASIC_IMG_SHELL
                        comma separated unit configuration
  --basic_nosqli BASIC_NOSQLI
                        comma separated unit configuration
  --basic_sqli BASIC_SQLI
                        comma separated unit configuration
  --cookies COOKIES     comma separated unit configuration
  --form_submit FORM_SUBMIT
                        comma separated unit configuration
  --git GIT             comma separated unit configuration
  --logon_cookies LOGON_COOKIES
```

(continues on next page)

(continued from previous page)

```
                        comma separated unit configuration
  --robots ROBOTS       comma separated unit configuration
  --spider SPIDER       comma separated unit configuration
  --unzip UNZIP         comma separated unit configuration
```

2.2 Configuration

Configuration parameters can either be set in an `.ini` file or at runtime via the `set` command. Configuration files are parsed using the built-in Python `configparser` module. The most important section is the `manager` section, which defines a few key parameters:

```
[manager]
# Flag format REGEX
flag-format=FLAG{.*?}
# Output directory
outdir=./results
```

Other parameters can be seen by running *set manager* at the katana prompt to receive a listing of the values currently set at runtime. When using the *set* command, parameters are specified with their fully qualified section/parameter name like so:

```
set manager[flag-format] NEWFLAG{.*?}
```

If the section name is not specified, a default value is added which will be used for any subsequent sections which request that value. This is particularly useful for a configuration such as `dict`, which can be specified once and will then apply to all units which require a dictionary like so:

```
set dict /path/to/rockyou.txt
```

You can also override the dictionary of a specific unit by specifying the unit as the section name:

```
set steghide[dict] /path/to/different/dict.txt
```

2.3 Evaluating Targets

The `target` command is used to view, start, and stop target evaluation. The `target add` sub-command will queue a target to begin analysis. The target specified can be a path name, URL, or raw data. Katana will create an abstract `Target` object and deduce the type of data passed to in intelligently:

```
katana - waiting - 0 units queued
 target add --help
Usage: target add [-h] target [...]

positional arguments:
  target      the target to evaluate

optional arguments:
  -h, --help  show this help message and exit

katana - waiting - 0 units queued
```

(continues on next page)

(continued from previous page)

```
 target add ./tests/cases/orchestra
[+] ./tests/cases/orchestra: queuing target
```

After adding a target, you can view the progress of all targets with the `target list` command:

```
katana - waiting - 0 units queued
 target list --help
Usage: __main__.py list [-h] [--completed] [--running] [--all] [--flags]

optional arguments:
  -h, --help       show this help message and exit
  --completed, -c  Display only completed targets
  --running, -r    Display only running targets
  --all, -a        Display all targets (running/completed)
  --flags, -f      D`

 katana - running - 0 units queued
 target list

./tests/cases/orchestra - completed
 hash: 2f0a02add67b58de837c7be054ae9e77
 flag: JHDCTF{strings}
```

When a target locates a flag, it will produce an asynchronous message to the screen identifying the unit and the flag which was found. The flag will also be copied to the primary clipboard:

```
katana - waiting - 0 units queued
 target ad
strings(./tests/cases/orchestra) - completed!
  JHDCTF{strings} - (copied)
katana - running - 0 units queued
 target ad
```

After a target has located flag(s), you can view the solution path for a target using the `target solution` command:

```
katana - waiting - 0 units queued
 target solution -r ./tests/cases/evil_ducky.jpg
steghide(./tests/cases/evil_ducky.jpg)
  strings(./results/60959e0ca0e4a202fd928c50f49a34fb/steghide/dGlua2Vy)
   JHDCTF{we_finally_found_the_the_flag} - (copied)
```

2.4 Monitoring Directories

The Katana REPL has the ability to utilize the `watchdog` Python module to monitor a directory or list of directories for new files and queue them for evaluation automatically. The allows you to start a Katana for a CTF, and then simply download interesting targets to a directory, checking periodically for flags or hung targets. The `monitor` command can be used to add, remove, and list monitored directories:

```
katana - waiting - 0 units queued
 monitor --help
Usage: monitor [-h] {list, ls, l, remove, rm, r, add, a} ...

Begin monitoring the given directory and automatically queue new targets as they are␣
↪created.
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit

subcommands:
  {list, ls, l, remove, rm, r, add, a}
                        Actions
    list
    remove (rm, r)      remove a monitored directory
    add (a)             begin monitoring a new directory
```

2.5 CTFd Integration

The Katana REPL has support to integrate with CTFd platforms. This integration includes the following:

- List challenges
- View challenge details (including solve state)
- Queue challenge (attached files and/or description)
- Automatically submit flags

This functionality is exposed through the `ctfd` command. All `ctfd` functions depend on a new configuration section named `ctfd`:

```
[ctfd]
url=http://ctfd.yourdomain.com
username=YourUserName
password=YourPassword
```

After you specify these configuration items, you can use the `ctfd list` command to list available challenges. The list is ordered from lowest-to-highest value, with solved challenges placed at the bottom. If your terminal supports extended escape sequences, solved challenges will be "dim" and struck-through:

```
katana - waiting - 0 units queued
 set ctfd
[ctfd]
url = http://192.168.1.37:8000
username = User01
password = password

katana - waiting - 0 units queued
 ctfd list
ID Title      Points
1  Orchestra  25
```

The `ctfd show` command will show the details of a given challenge ID:

```
katana - waiting - 0 units queued
 ctfd show 1
Orchestra - 25 points - solved

 It's music to my ears!
```

(continues on next page)

(continued from previous page)

```
 Files:
  - orchestra
```

To queue a challenge for evaluation, you can use the *ctfd queue* command. By default, this command only queues attached files. To also queue the description of the challenge for evaluation, use the `--description/-d` flag. It will also check that the given challenge is not already solved (although this can be bypassed with the `--force` flag):

```
katana - waiting - 0 units queued
 ctfd queue --force 1
[+] ctfd: queuing http://192.168.1.37:8000/files/f36fce4574bed199beb8170ac5b9bc1e/
↪orchestra?token=eyJ0ZWFtX2lkIjpudWxsLCJ1c2VyX2lkIjozLCJmaWxlX2lkIjoxfQ.Xbd3yA.
↪cKg9KcdqjStAQNAtHY5LP_m5uCw

strings(http://192.168.1.37:8000/files/f36fce4574bed199beb8170ac5b9bc...) - completed!
  JHDCTF{there_is_no_orchestra_without_the_strings} - (copied)

[+] ctfd: correct flag for challenge 1
```

In this case, automatic flag submission was turned on, and the flag was automatically submitted upon completion to CTFd. The updated `solved` state will be visible immediately in both `ctfd list` and `ctfd show`.

CHAPTER 3

Converting Units

When rewriting the Katana framework, a lot of changes were made to the *katana.unit.Unit* interface. We tried to keep the changes to a minimum, however some changes were inevitable. This should guide you through the changes in order to either write a new unit or convert an old one.

While most of the interface remains unchanged, a few new features were added. Specifically, the *katana.manager.Manager* class now takes the place of the old *Katana* object. *katana.target.Target* has largely been unchanged from the previous version.

3.1 Dependency Changes

All properties of a Unit are now contained within the class. The *DEPENDENCIES* variable is now a property, however it functions in the same capacity. The dependency mechanism can now be overridden through the *katana.unit.Unit.check_deps()* method, however this is almost never needed.

3.2 Groups

Units are now parts of *groups* which allow you to arbitrarily group units into logical sections. By default all converted units should be added to a group in conjunction with their package (e.g. "stego" or "crypto"). This allows old functionality like excluding those groups to remain. These group names can be specified when interfacing with the *katana.unit.Finder* class and therefore also with the Manager's `units` and `exclude` options. Those options can all either take a unit name or one of it's groups.

3.3 Recursion Preferences

The old *PROTECTED_RECURSE* and *RECURSE* properties have been changed and a new recursion protection mechanism is now in place. To modify recursion rules, you can now use the *katana.unit.Unit.RECURSE_SELF*, *katana.unit.Unit.NO_RECURSE* and *katana.unit.Unit.BLOCKED_GROUPS*.

BLOCKED_GROUPS allows you to outlaw recursion into entire groups of units. For example, you may outlaw recursion into any unit which is in the *crypto* group to prevent excessive recursion.

3.4 Reporting Data

The old data reporting mechanisms were part of the Katana class. In the new framework, these were moved to the *katana.manager.Manager* class and are all named as `register_*`. For example, to register arbitrary data as a result of this unit, you would call:

```
data = {"Wow": "This is really cool. FLAG{flag}"}
self.manager.register_data(self, data)
```

The manager will iterate through your data, and look for flags. It will also report the data to the *katana.monitor.Monitor*.

3.5 Generating and Reporting Artifacts

Artifact creation used to be handled by the Katana class, but has been moved to *katana.unit.Unit*. However, the interface is largely the same for creating an artifact. To create an artifact, use the *katana.unit.Unit.generate_artifact()* method. The interface and parameters are the same as the old *katana.generate_artifact* method. The biggest difference is that the artifact will not automatically be registered with the Manager or reported to the Monitor. To do that, call `katana.manager.Manager.register_artifact()`. As an example, if you have some data you think is a file:

```
data = b"Something that's probably a file!"
name, stream = self.generate_artifact("interesting", create=True)
stream.write(data)
stream.close()
self.manager.register_artifact(self, name)
```

CHAPTER 4

Module Reference

4.1 Manager - Evaluation Manager

A katana manager which is capable managing the evaluation of arbitrary units against an arbitrary number of Targets of varying types in a multithreaded manner and reporting results to a Monitor object

class katana.manager.**Manager**(*monitor: katana.monitor.Monitor = None*, *config_path=None*, *default_units=True*)

Class to manage the threaded evaluation of applicable units against arbitrary targets. Facilitates work queue management and recursion within given units. It will also manage output file creation (such as artifacts).

4.2 Monitor - Target Results Callback

class katana.monitor.**Monitor**

A monitor object recieves notifications from units whenever data, artifacts or flags are found while processing a target. The default monitor simply saves all artifacts to the artifact directory, and recurses on all data. It will also print flags to the console via it's logger.

on_artifact(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *path: str = None*) → None

Notify the monitor that an artifact was found and may be of interest to store in a file. This may be a temporary file already open (which will be lost after the unit ends) or some data which appears to be a file. By default, this file is saved under the *outdir* directory of the Manager. The return value indicates whether a new target should be queued for recursion with this artifact as an upstream

on_completion(*manager: katana.manager.Manager*, *timed_out: bool*) → None

This is called upon completion of evaluation (after manager.join() is complete). *timed_out* indicates if we reached a timeout.

on_data(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *data: Any*) → None

Notify the monitor of arbitrary data returned by a unit. The data could be of any type, but is likely *bytes* and should never be *str* (for complient units). The return value should indicate whether the given data should be recursed on (or re-evaluated for further unit processing). By default, all data is recursed on

on_depth_limit(*manager: katana.manager.Manager*, *target: katana.target.Target*, *unit: katana.unit.Unit*) → None
This means we reached the manager['manager']['max-depth'] limit during recursion.

on_download_update(*manager: katana.manager.Manager*, *download: katana.manager.Download*) → None
Called at most once per second while downloading files for targets

on_exception(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *exc: Exception*) → None
Notify the monitor that an exception occurred while processing a given unit. The exception is passed as the *exc* parameter

on_flag(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *flag: str*) → None
Notify the monitor that a flag was found

on_manager_exception(*manager: katana.manager.Manager*, *exc: Exception*) → None
Called when the manager catches an exception. By default, we do nothing. This is most likely a KeyboardInterrupt or some other signal that was sent to the main thread.

on_work(*manager: katana.manager.Manager*, *threadid: int*, *unit: katana.unit.Unit*, *case: Any*)
Keep track of the thread statuses for asynchronous status updates

class katana.monitor.**LoggingMonitor**(**args*, ***kwargs*)

on_artifact(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *path: str = None*) → None
Log a new artifact

on_exception(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *exception: Exception*) → None
Notify the monitor that an exception occurred while processing a given unit. The exception is passed as the *exc* parameter

on_flag(*manager: katana.manager.Manager*, *unit: katana.unit.Unit*, *flag: str*)
Log the solution chain of units which resulted in the given flag

class katana.monitor.**JsonMonitor**

on_completion(*manager: katana.manager.Manager*, *timed_out: bool*) → None
This is called upon completion of evaluation (after manager.join() is complete). *timed_out* indicates if we reached a timeout.

4.3 Unit - Abstract Challenge Solution

Units are the part of Katana which actually performs the evaluation on a given target. They are defined as subclasses of the `katana.unit.Unit` class, and must implement three methods at a minimum to begin functioning.

Units are automatically loaded from the *katana/units* directory, and optionally other directories specified at runtime by the *Finder* object below. This object is created by the Manager, and will search a directory for valid unit objects.

You can also register unit classes manually with the Finder if needed.

class katana.unit.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: `object`

Abstract the interface with a specific unit of evaluation for CTF challenges. This class must implement the *evaluate* and *validate* methods in order to be used with Katana.

Units attempt to solve very basic and targeted CTF challenges and provide data which could either contain a flag or another challenge. If the data contains a flag, evaluation will be halted. If it doesn't, the data may be used to bootstrap further Unit scanning.

When implementing a new unit, keep in mind that any serious processing should not occur until the `Unit.evaluate` method. This method is executed within the context of a thread. Executing intensive checks in other methods could indadvertedly slow down Katana.

Property PRIORITY This is a priority from 0 (highest priority) to 100 (lowest priority). Priorities are also scaled proportionally to parents in order to ensure children have higher priorities.

Property RECURSE_SELF Specifies whether this unit can recurse into itself.

Property NO_RECURSE Indicates if recursion is allowed at all for this unit

Property DEPENDENCIES A list of system binary dependencies we rely on (e.g. `["steghide"]`)

Property STRICT_FLAGS If specified, a flag must match the entire data string (not some substring within it).

Property GROUPS a list of groups this unit belongs to. This is useful for queuing or excluding only certain groups. By convention, this normally at least contains the package name (e.g. "crypto" or "stego"). However, it can theoretically contain any name you would like.

Property BLOCKED_GROUPS a list of groups or unit names which this unit cannot recurse into.

Here's an example of a very basic unit class:

```
class Unit(katana.unit.Unit):

    # Higher priority than normal
    PRIORITY = 25
    # Groups we belong to
    GROUPS = ["web", "bruteforce"]

    def __init__(manager: katana.manager.Manager, target: katana.target.Target):
        super(Unit, self).__init__(manager, target)
        if not target.is_url:
            raise NotApplicable("not a url")

    def evaluate(self, case):
        # Do something with this URL
        return
```

PRIORITY = 50

RECURSE_SELF = False

NO_RECURSE = False

PROTECTED_RECURSE = False

DEPENDENCIES = []

STRICT_FLAGS = False

GROUPS = []

BLOCKED_GROUPS = []

classmethod get_name() → str

By default, we assume the unit name is the same as the containing module. This can be overridden, but should not conflict with other units.

classmethod validate(*manager*)
Checks that required configuration values are available in the manager configuration file. This should be called via *super* prior to subclass implementation, as it ensures the section for this unit is added.

can_recurse(*unit_class: Type[katana.unit.Unit]*) → bool
Checks recursion rules and returns whether or not recursion is allowed into the given unit class. This unit has already been matched to a given recursion target from this unit.

Direct indicates whether this is the direct child or an ancestor of self.

Parameters **unit_class** – The child we are thinking recursing into

is_complete() → bool
Returns true if either this unit or the origin target has completed

enumerate()
Yield cases for evaluation given the target and manager configuration. This allows units with multiple possible evaluations (such as password guesser's) to take advantage of the parallelism of Katana without further coding. By default, this method yields a single *None* value which will be passed as *case* in the *Unit.evaluate* method below. You must yield at least one value before returning, or `evaluate` will never run.

evaluate(*case: Any*)
Run unit tasks given *case* which was returned from *Unit.enumerate*. This could happen in any thread or process of execution and should be stateless.

get_output_dir()
Find the output directory for this unit. This will return the directory where artifacts are expected to be stored in this context and also ensure it exists

generate_artifact(*name: Optional[str] = None*, *mode: str = 'w'*, *create: bool = True*, *asdir: bool = False*) → Tuple[str, IO]
Generate a new artifact, and return the path and open file handle. The artifact is not automatically registered with the manager, since it is initially empty. You should register any artifacts which contain useful data based on your unit (using `self.manager.register_artifact`)

family_tree() → Generator[Unit, None, None]
A generator which yields all parent units

get(*name: str*, *default: Optional[str] = None*) → str
Get a configuration value with a default. If a value was specified under the *DEFAULT* section, it will be returned before the default value specified here. :param name: name of the parameter :param default: default value :return: the value or the default value

getb(*name: str*, *default: Optional[bool] = None*) → bool
same as get, but returns a boolean value

geti(*name: str*, *default: Optional[int] = None*) → int
same as get but returns an integer value

classmethod check_deps()
The default dependency check will make sure that every item in self.DEPENDENCIES exists as an external executable in the current environment, and raise a NotApplicable exception otherwise. You likely won't need to override this, but you can if you'd like.

class katana.unit.**Finder**(*manager: katana.manager.Manager*, *use_default: bool = True*)
Bases: `object`

Utilize python dynamic introspection and loading to locate units either within the default unit list bundled with Katana or in a custom location.

Note: This code will automatically load all `*.py` scripts underneath the specified unit directory and look for a `Unit` class. This could be dangerous. Don't put random scripts in this directory.

validate() → None
Validate the manager configuration for each unit. Units without proper configuration will raise an exception which will be fed up to the user. Each unit accepts configuration items under it's own section if required (e.g. [katana.units.crypto.caeser])

find(*directory: str*, *prefix: str*) → Generator[Type[katana.unit.Unit], None, None]
Locate units which conform to the Katana unit specification with the given directory. All python source file within the directory will end up being executed. A valid unit definition contains a `Unit` class which subclasses `katana.unit.Unit` and implements `Unit.evaluate` and `Unit.validate`.

register(*unit: Type[katana.unit.Unit]*)
Register a unit to be used during analysis

match(*target: katana.target.Target*, *scale: float = 1.0*) → Generator[katana.unit.Unit, None, None]
Match the given target to one or more units that have previously been enumerated with the `Finder.find` method. This tests that the unit itself is applicable to the target in order to find specific applicable units

exception katana.unit.**NotApplicable**
Bases: `Exception`

Indicates the Unit which was created is not applicable to the given target, and the unit is in an undefined state.

exception katana.unit.**MissingDependency**
Bases: `Exception`

Indicates the unit was missing a dependency, and cannot be loaded. The message content is the name of the missing dependency

class katana.unit.**NoneUnit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.Unit*

classmethod get_name() → str
By default, we assume the unit name is the same as the containing module. This can be overridden, but should not conflict with other units.

class katana.unit.**FileUnit**(*manager: katana.manager.Manager*, *target: katana.target.Target*, *keywords=None*)
Bases: *katana.unit.Unit*

This unit base class requires that the given target be a file, and also optionally have a libmagic signature which contains one of a specified set of keywords. To use this unit, you simply pass a special *keywords* argument to its constructor in your unit subclass:

```
# A unit that requires a file containing some sort of image
class Unit(units.FileUnit):
    def __init__(self, manager, target):
        super(Unit, self).__init__(manager, target, keywords=['image'])
```

class katana.unit.**PrintableDataUnit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.Unit*

This unit base class ensures that the target content contains only printable data (that is, data which is not binary/is readable).

class katana.unit.**NotEnglishUnit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.Unit*

This unit base class ensures that the target content contains mostly non-english text.

class katana.unit.**NotEnglishAndPrintableUnit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.Unit*

This unit base class ensures that the target content is printable, and is also *not* english text (e.g. base64 data, white space, etc.)

class katana.unit.**RegexUnit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.Unit*

Utilizes a regular expression pattern to locate matching sections of the input data. The Unit will raise NotApplicable if the target has no matches.

enumerate()

Yield's all the match objects

4.4 Target - Abstract Challenge Data

The *Target* class abstracts away interactions with raw target data by first evaluating what kind target the data is, and providing convenience methods for accessing raw data, files, or URLs from the data.

class katana.target.**Target**(*manager: katana.manager.Manager*, *upstream: bytes*, *parent: Optional[katana.unit.Unit] = None*, *config: Optional[configparser.ConfigParser] = None*)

A Target has two main parts:

- Upstream
- Raw Data

The Upstream is what was passed to the target constructor. In the case of raw data, `upstream` and `raw` will be identical objects. If a URL was passed to the constructor, `raw` will take the form of the content of the web page. Katana will automatically attempt to fetch the page. In a similar fashion, `raw` will return the content of a file, if the upstream was a path.

If you don't rely on external tools, you should mostly deal with `raw` or `stream`. `raw` will either be a bytes object, or a memory mapped file (which acts like a bytes object in most situations). `stream` will either be an open file handle for file upstreams, or a BytesIO object which will act like a file. This allows you to reference the data in an abstract way no matter what the upstream target was. Other useful properties are also available which describe the data and are listed below.

Property upstream A bytes object holding the original target data.

Property parent A Unit object describing how this target was created (or None for root targets).

Property is_printable Whether the data is mostly printable text

Property is_english Whether the data appears to be mostly english

Property is_image Whether the data is an image

Property is_base64 Whether the data looks like base64

Property path The path to a file-backed target (URLs are also file-backed by an artifact)

Property completed Whether we are done processing this target

Property url_pieces A regex Match object containing the URL pieces, if this is a URL.

Property is_url True if this appears to be a valid URL

Property is_file True if this appears to be a valid file path. This is also true, if `manager[download]` is True, and we were able to download the file as an artifact.

Property magic libmagic result for the data

Property hash A hashlib.md5 object representing the hash of the data

Property start_time The time in seconds that this target was started

Property end_time When this target completed

Property units_evaluated The total number of units evaluated under this target (only root targets)

add_unit()
Add a unit for tracking. This is called by Manager.queue

build_target()
This method does the resource intensive part of building the target. It is done in a separate thread to decrease the time to return from the *Manager.queue_target* method (e.g. when running w/ a REPL)

is_webpage
Opposite of is_website_root?

is_website_root
if this is a URL, return whether we are at the root of the URL

raw
Return a bytes-like object for any given target type:

- Files/content already in memory: return self.content
- Files already written to disk: return a mmap object
- For all other unknown data: return self.upstream directly

rem_unit()
Remove a unit for tracking. Also sets completed if all units are done.

stream
Return a file-like object for any given target type:

- Files/content already in memory: return a BytesIO object
- Files already written to disk: return an binary file handle
- For all other unknown data: return a BytesIO object of upstream

web_host
if this is a URL, return the hostname

web_port
if this is a URL, return the port number

web_protocol
if this is a URL, return the protocol

web_query
if this is a url, return the query string

web_uri
if this is a url, return the URI

website_root
if this is a url, return the root of the URL (without any URI)

CHAPTER 5

katana.units.apk — Android Packages

These units handle procedures surrounding a given .apk file, or a downloaded Android package.

5.1 katana.units.apk.apktool — Decompile APK

Decompile an APK file with apktool.

This unit depends on the apktool external dependencies. It must be within your $PATH for Katana to use it properly.

All this unit does is call the command

```
apktool decode -f <the_target> -o <artifact_path>
```

It then looks through the results and queues each new file as targets to recurse on.

class katana.units.apk.apktool.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['apktool']

GROUPS = ['apk']

PRIORITY = 40

evaluate(*case: Any*) → None

This evaluate function calls the command:

```
apktool decode -f <the_target> -o <artifact_path>
```

and loops through the results, queuing each new file as a new target to recurse on.

Parameters **case** – A case returned by enumerate. For this unit, the enumerate function is not used.

Returns None. This function should not return any data.

CHAPTER 6

katana.units.crack — Hash Cracking

These units attempt to crack hashes, if they are ever found or determined in Katana's operations.

6.1 katana.units.crack.md5 — Crack MD5 Hash

Attempt to crack an MD5 hash.

This unit finds potential MD5 hashes matching the defined regular expression:

```
MD5_PATTERN = re.compile(rb"[a-fA-F0-9]{32}", re.DOTALL | re.MULTILINE)
```

This unit cracks the MD5 hash by using a supplied password or dictionary file. Currently it does not support reaching out to an online cracker, though this would be ideal.

class katana.units.crack.md5.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.Unit*

GROUPS = ['crack', 'bruteforce']

NO_RECURSE = True

PRIORITY = 75

enumerate() → Generator[Any, None, None]

Yield unit cases. This will read in the supplied password or a given dictionary file to generate new MD5 hashes and test them against the supplied MD5 hash target.

Returns Generator of target cases, in this case a byte string.

evaluate(*case: Any*) → None

Evaluate the target. This will take the current `case` supplied by the `enumerate` function, generate an MD5 hash with it and compare it to the supplied target. If it is a match, we have successfully cracked the hash and that `case` value is registered as new data.

Parameters **case** – A case returned by `enumerate`

Returns None. This function should not return any data.

CHAPTER 7

katana.units.crypto — Cryptography

These units handle procedures that are often necessary for challenges in the Cryptography category of CTFs.

Note: Often times, these units can take a long amount of time and bottleneck Katana's operations. If you know you do not need these checks, include `--exclude crypto` in your command.

Crypto units are often applicable to lots of targets, and considering they can do some brute-force operations, they often take up a lot of processing and can waste time for Katana's operations.

For this reason, we implemented a commonly used *katana.units.crypto.CryptoUnit* that checks to ensure the target is not a viable URL (to not clobber web units) and it is not a potentially useful file (like an image, document, or something else specific).

class katana.units.crypto.**CryptoUnit**(**args*, ***kwargs*)
This Unit will raise *katana.unit.NotApplicable* if the unit is a URL or a potentially useful file.

7.1 katana.units.crypto.affine — Affine Cipher

Attempt to decrypt a target with the classic Affine cipher.

You can read more about the Affine cipher here: https://en.wikipedia.org/wiki/Affine_cipher

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English.

You can supply and customize the given A and B values as well as the alphabet to be used in the Affine cipher operation, though by default this will bruteforce and use the range provided with the English alphabet, letters A-Z.

class katana.units.crypto.affine.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
This unit does not recurse into other Crypto units because that might spiral into a disaster.

GROUPS = ['crypto', 'affine']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name of the unit, "affine".

PRIORITY = 65
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat lower priority due to how uncommon this is within CTFs.

RECURSE_SELF = False
This unit should not recurse into itself. That could spiral in to an infinite loop.

enumerate() → Generator[Any, None, None]
Yield unit cases. This will check if any given `A` or `B` values are supplied to the unit. If a value is not supplied, it will use all numbers up the length of the alphabet (which can also be supplied), by default, the English letters A-Z. The corresponding value will be the greatest common denominator between that in the length, as that is the only correspondent value that is mathematically required for the Affine cipher to work.

Returns Generator of target cases, in this case a tuple of `A` and `B` values.

evaluate(*case: Any*) → None
Evaluate the target. This will perform

Parameters **case** – A case returned by `enumerate`, in this case a tuple of `A` and `B` values.

Returns None. This function should not return any data.

katana.units.crypto.affine.**affine**(*c: int*, *a: int*, *b: int*, *alphabet: bytes*) → str
Perform the affine cipher for a single letter.

C An integer value for the given letter (its location within the alphabet)

A An integer value for the A value used in the Affine cipher operation.

B An integer value for the B value used in the Affine cipher operation.

Alphabet A bytes string for the supplied alphabet.

7.2 katana.units.crypto.atbash — Atbash Cipher

Perform the classic Atbash cipher on the given target.

You can read more about the Atbash cipher here: https://en.wikipedia.org/wiki/Atbash

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

The gist of the Atbash cipher is that it will perform a substitution cipher with the key being the typical English alphabet, just reversed. Basically, A-Z maps to Z-A.

class katana.units.crypto.atbash.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
This unit does not recurse into other Crypto units because that might spiral into a disaster.

GROUPS = ['crypto', 'atbash']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name of the unit, "atbash".

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat lower priority due to how uncommon this is within CTFs.

RECURSE_SELF = False
This unit should not recurse into itself. That could spiral in to an infinite loop.

evaluate(*case: Any*) → None
This `evaluate` function performs the Atbash cipher on the target.

Parameters **case** – A case returned by `enumerate`. For this unit, the enumerate function is not used.

Returns None. This function should not return any data.

7.3 katana.units.crypto.caesar255 — Caesar Cipher with ASCII 255

Perform a Caesar cipher, with the key mapping in the range of all 255 ASCII characters, on the target.

You can read more about the Caesar cipher here: https://en.wikipedia.org/wiki/Caesar_cipher

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

class katana.units.crypto.caesar255.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
This unit does not recurse into other Crypto units because that might spiral into a disaster.

GROUPS = ['crypto', 'caesar255']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name of the unit, "caesar255".

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty.

RECURSE_SELF = False
This unit should not recurse into itself. That could spiral in to an infinite loop.

enumerate() → Generator[Any, None, None]
Yield unit cases. The end-user can either supply a `shift` value as an argument, or it will bruteforce all the possible shift values within in the ASCII range (i.e. try the numbers 1-255).

Returns Generator of target cases, in this case an integer for the shift value (provided, or range 1-255).

evaluate(*shift: int*) → None
Perform the caesar cipher on the target.

Parameters **case** – A case returned by `enumerate`, in this case, the shift value to use for the Caesar Cipher operation.

Returns None. This function should not return any data.

7.4 katana.units.crypto.caesar — Caesar Cipher with 26 Letters

Perform a Caesar cipher on the target.

You can read more about the Caesar cipher here: https://en.wikipedia.org/wiki/Caesar_cipher

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

class katana.units.crypto.caesar.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']

This unit does not recurse into other Crypto units because that might spiral into a disaster.

GROUPS = ['crypto', 'caesar']

These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name of the unit, "caesar".

PRIORITY = 40

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat higher priority due to how common this is within CTFs.

RECURSE_SELF = False

This unit should not recurse into itself. That could spiral in to an infinite loop.

enumerate() → Generator[Any, None, None]

Yield unit cases. The end-user can either supply a `shift` value as an argument, or it will bruteforce all the possible shift values within in the English alphabet (i.e. try the numbers 1-25).

Returns Generator of target cases, in this case an integer for the shift value (provided, or range 1-25).

evaluate(*case: Any*) → None

Perform the caesar cipher on the target.

Parameters **case** – A case returned by `enumerate`, in this case, the shift value to use for the Caesar Cipher operation.

Returns None. This function should not return any data.

katana.units.crypto.caesar.**shift_char**(*c: str*, *shift: int*, *alphabet: str*) → str

This is a convenience function that will perform the most primitive operation of the Caesar Cipher – shifting one character by a given amount within the given alphabet.

7.5 katana.units.crypto.dna — T,A,C,G,U DNA Letters

DNA/Codon Cipher.

This unit will translate groupings of letters (A,T,C,G,U) into 22 out of 26 possible English characters.

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

class katana.units.crypto.dna.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
This unit does not recurse into other Crypto units because that might spiral into a disaster.

GROUPS = ['crypto', 'dna']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name of the unit, "dna".

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty.

RECURSE_SELF = False
This unit should not recurse into itself. That could spiral in to an infinite loop.

evaluate(*case: Any*) → None
Evaluate the target.

Read individual Codon groupings and replace them with the corresponding English character.

Parameters **case** – A case returned by `enumerate`. In this case, the `enumerate` function is not used.

Returns None. This function should not return any data.

7.6 katana.units.crypto.phonetic — NATO Phonetic Alphabet

class katana.units.crypto.phonetic.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.RegexUnit*

BLOCKED_GROUPS = ['crypto']
This unit does not recurse into other Crypto units because that might spiral into a disaster.

GROUPS = ['crypto', 'phonetic']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name of the unit, "phonetic".

PATTERN = regex.Regex(b'(alfa|alpha|bravo|charlie|delta|echo|foxtrot|golf|hotel|india|
This pattern is used specifically for this unit to detect the NATO phonetic alphabet.

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty.

evaluate(*match: re.Match*) → None
Evaluate the target.

Parameters **match** – A single regular expression match

Returns None

7.7 katana.units.crypto.polybius — Polybius Square Cipher

Attempt to decrypt a Polybius Square cipher.

You can read more about the Polybius Square cipher here: https://en.wikipedia.org/wiki/Polybius_square

This unit will search for numbers and translate them to the proper mapping within a Polybius square.

class katana.units.crypto.polybius.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.RegexUnit*

GROUPS = ['crypto', 'polybius']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, as well as the name for this unit.

PATTERN = regex.Regex(b'([1-5]+ ?)+', flags=regex.A | regex.S | regex.M | regex.V0)
This pattern is used specifically for this unit to detect the data used for the Polybius cipher.

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a default priority.

evaluate(*match*) → None
Evaluate the target.

Parameters **match** – A single regular expression match. In this case, this should retrieve numbers to be used to map to letters within the Polybius Square.

Returns None. This function should not return any data.

7.8 katana.units.crypto.quipqiup — Online Substitution Cipher solver

Substituion cipher solver, by outsourcing to https://quipqiup.com/.

The gist of this code is ripped from https://github.com/rallip/substituteBreaker. The unit takes the target, and if it does not look English text but it is clearly printable characters, it offers it to quipqiup online.

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

Note: This unit **does not recurse**. It simply looks for flags in the output of quipqiup's best potential solution. Note that Katana might find flags that are not in the specific flag format, but also denoted in a "the flag is:" structure.

---

class katana.units.crypto.quipqiup.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'quipquip', 'substitution']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit and some other related topics.

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a slightly lower priority.

RECURSE_SELF = False
This unit **does not recurse**. It simply looks for flags in the output of quipqiup's best potential solution.

evaluate(*case: Any*) → None
Evaluate the target.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

`katana.units.crypto.quipqiup.`**decodeSubstitute**(*cipher: str*, *time=3*, *spaces=True*) → str

This is stolen from https://github.com/rallip/substituteBreaker All it does is use the `requests` module to send the ciphertext to quipqiup and returns the results as a string.

7.9 `katana.units.crypto.railfence` — Railfence Cipher

Railfence Cipher decoder

This takes arguments `rails` and `offset` which you can set, but they will be bruteforce within the range of 2-10 and 0-10 respectively.

This unit inherits from the *`katana.unit.NotEnglishAndPrintableUnit`* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *`katana.units.crypto.CryptoUnit`* class to ensure it is not a viable URL or potentially useful file.

The code for this is shamelessly stolen from https://github.com/tothi/railfence

class `katana.units.crypto.railfence.`**Unit**(**args*, ***kwargs*)

Bases: *`katana.unit.NotEnglishAndPrintableUnit`*, *`katana.units.crypto.CryptoUnit`*

BLOCKED_GROUPS = ['crypto']

These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'railfence']

These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit, "railfence".

PRIORITY = 60

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a slightly lower priority.

RECURSE_SELF = False

This unit does not recurse into itself. That would be silly.

enumerate()

Yield cases for evaluation given the target and manager configuration. This allows units with multiple possible evaluations (such as password guesser's) to take advantage of the parallelism of Katana without further coding. By default, this method yields a single *None* value which will be passed as *case* in the *Unit.evaluate* method below. You must yield at least one value before returning, or `evaluate` will never run.

evaluate(*case: Any*) → None

Evaluate the target. This simply attemptes to decrypt the target with the Railfence cipher, using the `rails` and `offset` values returned by *enumerate*'.

Parameters **case** – A case returned by `enumerate`. In this case, it is a tuple containing a rail value and offset value to be used for the Railfence cipher operations.

Returns None

`katana.units.crypto.railfence.`**decryptFence**(*cipher*, *rails*, *offset=0*)

Stolen from https://github.com/tothi/railfence.

This is a convenience function to decrypt data with the Railfence cipher.

Parameters

- **cipher** – The ciphertext as a string.
- **rails** – The integer number of rails to use in the Railfence cipher operations.
- **offset** – The integer offset number to use in the Railfence cipher operations.

katana.units.crypto.railfence.**encryptFence**(*plain*, *rails*, *offset=0*)

Stolen from https://github.com/tothi/railfence.

This is a convenience function to encrypt data with the Railfence cipher.

Parameters

- **plain** – The plaintext as a string.
- **rails** – The integer number of rails to use in the Railfence cipher operations.
- **offset** – The integer offset number to use in the Railfence cipher operations.

7.10 katana.units.crypto.reverse — Simple Reverse

Reverse ciphertext

This will simply reverse the text and look for a flag.

This unit inherits from the *katana.unit.NotEnglishUnit* class, as we can expect the data to not be readable English (if it is in fact reverse text). It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

class katana.units.crypto.reverse.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.NotEnglishUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']

These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'reverse']

These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included.

PRIORITY = 70

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a lower priority.

RECURSE_SELF = False

Do not recurse into self

evaluate(*case: Any*) → None

Evaluate the target. This simply reverses the target.

Parameters **case** – A case returned by enumerate. In this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

7.11 katana.units.crypto.rot47 — ROT47 Cipher

ROT47 decoder

The gist of this code is ripped from https://rot47.net/_py/rot47.txt.

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

class katana.units.crypto.rot47.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'rot47']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit, "rot47".

PRIORITY = 45
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a slightly higher priority.

RECURSE_SELF = False
Do not recurse into self

do_rot47(*s*)
Shamelessly stolen from https://rot47.net/_py/rot47.txt

This function takes a string and performs the ROT47 operation on it.

Parameters **s** – The byte string to perform the ROT47 operation on.

evaluate(*case: Any*) → None
Evaluate the target.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None

7.12 katana.units.crypto.rsa — Attempt to solve RSA

RSA decryptor

This takes arguments "e", "n", "q", "p", "dq", "dp", "d", "c", "phi", though they will potentially be automatically decoded by the program if a plaintext file is supplied.

class katana.units.crypto.rsa.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.NotEnglishUnit*

BLOCKED_GROUPS = ['crypto']
These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'rsa']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit, "rsa".

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a slightly lower priority.

RECURSE_SELF = False
Do not recurse into self

evaluate(*case: Any*) → None
Evaluate the target.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should return any data.

katana.units.crypto.rsa.**contfrac_to_rational**(*frac: list*)
This function is used for the Weiner's Little D attack.

Converts a finite continued fraction `[a0, ..., an]` to an x/y rational.

katana.units.crypto.rsa.**convergents_from_contfrac**(*frac: list*) → list
This function is used for the Weiner's Little D attack.

Computes the list of convergents using the list of partial quotients

Parameters **frac** – Fractions represented by a list

Returns A list of convergents

katana.units.crypto.rsa.**egcd**(*a*, *b*)
This function is used for the Weiner's Little D attack.

Determines the Euclidean Greatest Common Denominator between given values.

Parameters

- **a** – One value to be used to find the GCD for.
- **b** – Another value to be used to find the GCD for.

Returns

katana.units.crypto.rsa.**find_cube_root**(*n*)
This function is used for the Cube Root attack.

Determines the cube root of a number.

Parameters **n** – The number to determine the cube root of.

Returns The resulting cube root.

katana.units.crypto.rsa.**find_variables**(*text*)
This is used to detect variables in a given file, or handle a given pubkey.

Parameters **text** – The string to pull the variables from.

Returns A Generator for an RSA letter variable and its value.

katana.units.crypto.rsa.**isqrt**(*n*)
This function is used for the Weiner's Little D attack.

Determines the integer square root of a nunber.

Parameters **n** – The number to determine the integer square root of.

Returns The resulting integer square root.

katana.units.crypto.rsa.**mod_inv**(*a*, *m*)
This function is used for the Weiner's Little D attack.

Deterine the modular inverse, given a base and the modulus.

Parameters

- **a** – The base to use for the modular inverse operation.

- **m** – The modulus to use for the modular inverse operation.

Returns An integer as the result of the modular inverse.

katana.units.crypto.rsa.**parse_int**(*given*)
This function will parse out a Python value regardless of the representation a number is given in the provided string. It will detect hex or an integer form.

Parameters **given** – The string information that potentially includes a number.

Returns The Python integer value found.

katana.units.crypto.rsa.**rational_to_contfrac**(*x: int*, *y: int*) → list
This function is used for the Weiner's Little D attack.

Converts a rational x/y fraction into a list of partial quotients [a0, . . . , an]

Parameters

- **x** – The numerator of the provided fraction.
- **y** – The denominator of the provided fraction.

Returns a list of partial quotients.

katana.units.crypto.rsa.**weiners_little_d**(*e*, *n*)
This function is used for the Weiner's Little D attack.

Actually

Parameters

- **e** – The RSA e-value (exponent).
- **n** – The RSA N-value (modulus).

Returns The determined RSA d-value (private key) after the Weiner's Little D attack.

7.13 katana.units.crypto.t9 — T9 phone keypad cipher

T9 Phone keypad Cipher

This unit will decode a T9 cipher and look for flags. This code relies on there being spaces between the T9 ciphers. It can be made cleaner with some regular expression processing, but it has not yet been done. . .

class katana.units.crypto.t9.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.RegexUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 't9']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit, "t9".

PATTERN = regex.Regex(b'[0-9*]+(\\w([0-9*]+))*', flags=regex.A | regex.V0)

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a defualt priority.

RECURSE_SELF = False
Do not recurse into self

decode_one(*number*)

evaluate(*match*)
Run unit tasks given *case* which was returned from *Unit.enumerate*. This could happen in any thread or process of execution and should be stateless.

7.14 katana.units.crypto.vigenere — Vigenere Cipher

Attempt to decrypt a Vigenere cipher.

You can supply a `key` argument to use for the Vigenere cipher operation. With the current implementation, if the key is not provided, this unit does not run (it does not attempt to bruteforce it or determine a key on its own).

This unit inherits from the *katana.unit.NotEnglishAndPrintableUnit* class, as we can expect the data to still be printable characters (letters, numbers and punctuation) but not readable English. It also inherits from the *katana.units.crypto.CryptoUnit* class to ensure it is not a viable URL or potentially useful file.

class katana.units.crypto.vigenere.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.NotEnglishAndPrintableUnit*, *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'vigenere']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit, "vigenere".

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a slightly lower priority.

RECURSE_SELF = False
Do not recurse into self.

evaluate(*case: Any*) → None
Evaluate the target.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.crypto.vigenere.**vigenere**(*plaintext*, *key*)
Perform a vigenere cipher.

Parameters

- **plaintext** – The plaintext message to use for the Vigenere cipher.
- **key** – The key to use for the Vigenere cipher.

Returns The resulting ciphertext from the Vignere cipher operation

7.15 katana.units.crypto.xor — XOR Operation

XOR decoder

You can supply a `key` argument to use for the XOR operation. With the current implementation, if the key is not provided, this unit will attempt to bruteforce the XOR with a single-byte range (1-255).

class katana.units.crypto.xor.**Unit**(**args*, ***kwargs*)
Bases: *katana.units.crypto.CryptoUnit*

BLOCKED_GROUPS = ['crypto']
These are tags for groups to not recurse into. Recursing into other crypto units would be silly.

GROUPS = ['crypto', 'xor']
These are "tags" for a unit. Considering it is a Crypto unit, "crypto" is included, and the name of the unit itself, "xor".

PRIORITY = 70
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a lower priority.

RECURSE_SELF = False
Do not recurse into self.

evaluate(*case: Any*) → None
Evaluate the target. Perform the XOR operation with the provided `key` argument. If no key is provided, it will bruteforce a single-byte XOR within the range of 1-255.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.crypto.xor.**xor**(*data*, *key*)
Perform an XOR operation across the provided data with a given key.

Parameters

- **data** – A byte string to use as the data for the XOR operation.
- **key** – The key to use the for the XOR operation.

Returns The result of the XOR operation as a byte string.

CHAPTER 8

katana.units.esoteric — Esoteric Languages

These units evaluate code that seems to be from a given esoteric language.

Note: Often times, these units can take a long amount of time and bottleneck Katana's operations. If you know you do not need these checks, include `--exclude esoteric` in your command.

8.1 katana.units.esoteric.brainfuck — Brainfuck

Unit for brainfuck esoteric language.

Given target data, this unit will ignore everything that is NOT valid Brainfuck characters and exclude them.

This unit includes a `evaluate_brainfuck` function that is often used by other units like Ook and Pikalang.

class katana.units.esoteric.brainfuck.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.Unit*

GROUPS = ['esoteric', 'brainfuck']

These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "brainfuck".

PRIORITY = 50

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a defualt priority.

evaluate(*case: Any*) → None

Evaluate the target. Run the target as Brainfuck code and give the standard output results to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.esoteric.brainfuck.**buildbracemap**(*code: bytes*) → dict
This is used for the Brainfuck operations. It will match opening and closing braces for use within the Brainfuck program.

Parameters **code** – A byte string of the Brainfuck code.

Returns a bracemap dictionary

katana.units.esoteric.brainfuck.**cleanup**(*code: bytes*) → str
This is used for the Brainfuck operations. It will clean the provided code to only find the appropriate Brainfuck operators.

Parameters **code** – A byte string of the Brainfuck code.

Returns Only the bytes of appropriate Brainfuck operators.

katana.units.esoteric.brainfuck.**evaluate_brainfuck**(*code: bytes*, *input_file*, *timeout: int = 1*)
This function actually runs the provided Brainfuck operations and returns the standard output.

Parameters

- **code** – The code to run as Brainfuck.
- **input_file** – A file to for the Brainfuck program to read as standard input. If this is not provided, it will yield a newline.
- **timeout** – A timeout value in seconds. After this time has elapsed, the Brainfuck code will stop executing.

Returns The standard output for the Brainfuck program.

8.2 katana.units.esoteric.cow — COW

Unit for Cow esoteric language.

Given target data, this unit will ignore everything that is NOT valid Cow characters and exclude them.

class katana.units.esoteric.cow.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.Unit*

GROUPS = ['esoteric', 'cow']
These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "cow".

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a defualt priority.

RECURSE_SELF = False
There is no reason to recurse into yourself. We shouldn't get cow out.

evaluate(*case: Any*) → None
Evaluate the target. Run the target as Cow code and give the standard output results to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.esoteric.cow.**build_jumpmap**(*code: bytes*) → dict
This is used for the Cow operations. It will match opening and closing braces for use within the Cow program.

Parameters **code** – A byte string of the Cow code.

Returns a jumpmap dictionary

katana.units.esoteric.cow.**cleanup**(*code: bytes*) → bytes
This is used for the Cow operations. It will clean the provided code to only find the appropriate Cow operators.

Parameters **code** – A byte string of the Cow code.

Returns Only the bytes of appropriate Cow operators.

katana.units.esoteric.cow.**evaluate_cow**(*code*, *input_file*, *timeout=-1*)
This function actually runs the provided Cow operations and returns the standard output.

Parameters

- **code** – The code to run as Cow.
- **input_file** – A file to for the Cow program to read as standard input. If this is not provided, it will yield a newline.
- **timeout** – A timeout value in seconds. After this time has elapsed, the Cow code will stop executing.

Returns The standard output for the Cow program,

8.3 katana.units.esoteric.jsfuck — JSFuck

JSFuck decoder

This unit will attempt to execute JSFuck and look for flags in the result.

class katana.units.esoteric.jsfuck.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.NotEnglishUnit*

DEPENDENCIES = ['node']
Because this requires JavaScript code, `node` is a necessary binary dependency.

GROUPS = ['esoteric', 'jsfuck', 'javascript']
These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "jsfuck", and the tag "javascript".

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately low priority because it requires an external tool.

RECURSE_SELF = False
It would not make sense to recurse into ourself. We shouldn't get JSFuck out.

evaluate(*case: Any*)
Evaluate the target. Run the target as JSFuck code and return the evaluated source code to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

8.4 katana.units.esoteric.malbolge — Malbolge

Unit to run code in the Malbolge esoteric language.

This code is shamelessly stolen from https://github.com/kmyk/malbolge-interpreter. We do not claim to know everything that it does... it is Malbolge, after all.

class katana.units.esoteric.malbolge.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.NotEnglishUnit*

GROUPS = ['esoteric', 'malbolge']

These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "malbolge".

PRIORITY = 10

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority.

evaluate(*case: Any*)

Evaluate the target. Run the target as Malbolge code and return the standard output to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.esoteric.malbolge.**crypt1**(*i*, *m*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**crypt2**(*m*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**crz**(*xs*, *ys*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**decrypt1**(*i*, *c*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**execute**(*code*, *inf=<_io.BufferedReader name='<stdin>'>*, *allow_not_isprint=False*, *debug=False*)

This function is execute Malbolge code.

katana.units.esoteric.malbolge.**execute_step**(*a*, *c*, *d*, *mem*, *inf=<_io.BufferedReader name='<stdin>'>*, *outf=<_io.BufferedWriter name='<stdout>'>*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**initial_memory**(*code*, *allow_not_isprint=False*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**isword**(*x*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**rotr**(*x*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**tri**(*x*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**unword**(*x*)

This function is used as part of Malbolge's operations.

katana.units.esoteric.malbolge.**word**(*ys*)

This function is used as part of Malbolge's operations.

8.5 katana.units.esoteric.ook — Ook

Unit for the Ook esoteric language.

This unit will map the Ook operations to their Brainfuck equivalant, and then pass along the actual execution to the Brainfuck unit's `evaluate_brainfuck` function.

class katana.units.esoteric.ook.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.NotEnglishUnit*

GROUPS = ['esoteric', 'ook']

These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "ook".

PRIORITY = 50

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has default priority

evaluate(*case: Any*) → None

Evaluate the target. Run the target as Ook code and give the standard output results to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.esoteric.ook.**evaluate_ook**(*code*, *input_file*, *timeout=1*)

This function will actually evaluate the Ook code, by translating it to Brainfuck character mapping and then passing it to the `evaluate_brainfuck` unit.

This function also verifies that the Ook code is not an odd-length string. That would result in improper Ook code.

Parameters

- **code** – A byte string of the Ook code.
- **input_file** – A file to for the Ook program to read as standard input. If this is not provided, it will yield a newline.
- **timeout** – A timeout value in seconds. After this time has elapsed, the Ook code will stop executing.

Returns The standard output for the Ook program,

8.6 katana.units.esoteric.piet — Piet

Piet esoteric language

This unit inherits from the *katana.unit.FileUnit* to ensure that the target is in fact an image file.

This unit will extract the text returned by a given Piet language image using the `npiet` command-line utility. The syntax is:

```
npiet -e 1000000 <target_path>
```

class katana.units.esoteric.piet.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['npiet']
Required depenencies for this unit "npiet"

GROUPS = ['esoteric', 'npiet', 'piet']
These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "npiet".

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = False
It would not make sense to recurse into ourself

evaluate(*case: Any*)
Evaluate the target. Run the `npiet` code and give the standard output results to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

8.7 katana.units.esoteric.pikalang — Pikalang

Pikalang esoteric decoder

This unit will map the Pikalang operations to their Brainfuck equivalant, and then pass along the actual execution to the Brainfuck unit's `evaluate_brainfuck` function.

In the previous framework of Katana, this unit attempted to decode Pikalang in seemingly TWO different variations. One was a literal mapping to Brainfuck code, the other did something different that required much more code (https://github.com/joelsmithjohnson/pikachu-interpreter)

I have not translated that other code to use bytes, and I do not see the need to do so currently, considering how obscure Pikalang is to begin with.

class katana.units.esoteric.pikalang.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.PrintableDataUnit*

GROUPS = ['esoteric', 'pikalang']
These are "tags" for a unit. Considering it is a Esoteric unit, "esoteric" is included, as well as the unit name "pikalang".

PRIORITY = 40
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. Has a slightly higher priority

evaluate(*case: Any*)
Evaluate the target. Run the target as Pikalang code and give the standard output results to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

CHAPTER 9

katana.units.forensics — Forensics

These units handle procedures that are often necessary for challenges in the Forensics category of CTFs.

9.1 katana.units.forensics.binwalk — Binwalk

Binwalk file carving

This unit will run `binwalk` to extract other files out of one given file. The syntax runs as:

```
binwalk -e <target_path> --directory <binwalk_directory> --dd=.* -M
```

class katana.units.forensics.binwalk.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['carver']

Groups which this unit cannot recurse into.

DEPENDENCIES = ['binwalk']

Required depenencies for this unit "binwalk". This must be in your PATH to be executed.

GROUPS = ['forensics', 'binwalk', 'carver']

These are "tags" for a unit. Considering it is a Forensics unit, "forensics" is included, as well as the unit name "binwalk".

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = False

Don't recurse into any of the extract objects. Binwalk should have carved them out already.

evaluate(*case: Any*)

Evaluate the target. Run `binwalk` on the target and recurse on any new found files.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.forensics.binwalk.**md5sum**(*path: str*) → _hashlib.openssl_md5
Quick convenience function to get the MD5 hash of a file

9.2 katana.units.forensics.foremost — Foremost

Binwalk file carving

This unit will run `foremost` to extract other files out of one given file. The syntax runs as:

```
foremost <target_path> -o <foremost_directory>
```

class katana.units.forensics.foremost.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['carver']
Groups which this unit cannot recurse into.

DEPENDENCIES = ['foremost']
Required depenencies for this unit "foremost". This must be in your PATH to be executed.

GROUPS = ['forensics', 'foremost', 'carver']
These are "tags" for a unit. Considering it is a Forensics unit, "forensics" is included, as well as the unit name "foremost".

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = False
Don't recurse into any of the extract objects. Binwalk should have carved them out already.

evaluate(*case: str*)
Evaluate the target. Run `foremost` on the target and recurse on any new found files.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.forensics.foremost.**md5sum**(*path*)
Quick convenience function to get the MD5 hash of a file

CHAPTER 10

katana.units.gzip — GZIP files

These units handle procedures that to work with GZIP archive files.

Admittedly, this should be bundled to in a larger, "archive" unit package, but this has not yet been done.

10.1 katana.units.gzip.gunzip — Extract GZIP Archive

GZIP file extraction

This unit works via the built-in Python library `gzip`, so there is no need for an external binary dependency.

The unit inherits from *katana.unit.FileUnit* to ensure the target is a GZIP file.

Note that GZIP files do not have support for passwords, so that is not implemented here.

class katana.units.gzip.gunzip.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

GROUPS = ['gzip', 'archive']

These are "tags" for a unit. Considering it is a GZIP unit, "gzip" is included, as well as the tag "archive".

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = True

This unit can recurse into itself because we can end up with nested GZIPS.

evaluate(*case: str*)

Evaluate the target. Extract the target with GZIP and recurse on any new found files.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

CHAPTER 11

katana.units.ocr — Optical Character Recognition

These units perform optical character recognition, to determine information that might only be displayed in images.

11.1 katana.units.ocr.tesseract — Tesseract

Unit to perform Optical Character Recognition with Tesseract.

The unit inherits from *katana.unit.FileUnit* to ensure the target is an image.

This unit uses the Python library for Tesseract, which must be installed for this to run.

class katana.units.ocr.tesseract.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

GROUPS = ['ocr', 'tesseract']

These are "tags" for a unit. Considering it is a Ocr unit, "ocr" is included, as well as the unit name "tesseract".

PRIORITY = 25

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority because this is lightweight.

RECURSE_SELF = False

Do not recurse into itself, since it will not provide another image.

evaluate(*case: Any*) → None

Evaluate the target. Attempt OCR on the target and recurse on any newfound data.

Parameters **case** – A case returned by enumerate. For this unit, the enumerate function is not used.

Returns None. This function should not return any data.

katana.units.ocr.tesseract.**attempt_ocr**(*image_path: str*) → str

Run tesseract against an image file and return the string found

Parameters **image_path** – The path to an image file.

Returns The string determined by Tesseract's OCR efforts.

CHAPTER 12

katana.units.pcap — Packet Capture Processing

These units process .pcap files and run other commands and tools that relate to them.

12.1 katana.units.pcap.tcpflow — tcpflow

tcpflow

This unit will carve out files from a given PCAP file using the tcpflow command-line utility. The syntax runs as:

```
tcpflow -r <target_path> -o <tcpflow_directory>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a PCAP file.

class katana.units.pcap.tcpflow.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['tcpflow']

Required depenencies for this unit "tcpflow"

GROUPS = ['network', 'pcap', 'tcpflow']

These are "tags" for a unit. Considering it is a pcap unit, "pcap" is included, as well as the tag "network", and unit name "tcpflow"

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = True

In case we have extract other PCAPs for some reason, we CAN recurse into ourselves.

evaluate(*case: Any*)

Evaluate the target. Run tcpflow on the target and recurse on any new found files.

Parameters **case** – A case returned by enumerate. For this unit, the enumerate function is not used.

Returns None. This function should not return any data.

CHAPTER 13

katana.units.pdf — PDF File Processing

These units process `.pdf` files and run other commands and tools that relate to them.

13.1 katana.units.pdf.pdf2text — pdf2text

Convert PDF to Text

This unit retrieves the text included in a PDF document, using the "pdftotext" Python library.

The unit inherits from *katana.unit.FileUnit* to ensure the target is a PDF file.

class katana.units.pdf.pdf2text.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['pdf']

PDFs shouldn't come out of this. So no reason to look.

GROUPS = ['pdf', 'pdftotext', 'pdf2text']

These are "tags" for a unit. Considering it is a pdf unit, "pdf" is included, and the name of the unit, "pdftotext"

PRIORITY = 25

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority if this is detected...

RECURSE_SELF = False

Again no PDF from this. So recursion is silly.

evaluate(*case: Any*) → None

Evaluate the target. Extract the text out of the PDF document and recurse on any newfound text.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

13.2 katana.units.pdf.pdfcrack — PDFCrack - Crack Password

Crack a password-protected PDF

This unit attempt to unlock a password-protected PDF file. This is done with the *PyPDF2* module in Python, which must be installed for this work. First the unit will try with an empty password, and then it will try with the user-supplied password argument. Finally, it will bruteforce with a supplied dictionary file.

The unit inherits from *katana.unit.FileUnit* to ensure the target is a PDF file.

Note: Note that it only (potentially) determines the password, but does nothing else with the file.

class katana.units.pdf.pdfcrack.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['pdf']
PDFs shouldn't come out of this. So no reason to look.

GROUPS = ['pdf', 'pdfcrack']
These are "tags" for a unit. Considering it is a pdf unit, "pdf" is included.

PRIORITY = 25
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority if this is detected...

RECURSE_SELF = False
Again no PDF from this. So recursion is silly.

enumerate()
This function will first yield an empty password, then the supplied password argument, then loop through each line of a provided dictionary file. The password will then be used by the `evaluate` function to try and open the encrypted PDF.

evaluate(*case: Any*) → None
Evaluate the target. Attempt to open the PDF document with a supplied password given by `enumerate`.

Parameters **case** – A case returned by `enumerate`. In this case, this will be a string value supplied as an argument or bruteforce via a supplied dictionary file.

Returns None. This function should not return any data.

13.3 katana.units.pdf.pdfimages — pdfimages - Extract Images

Extract PDF images

This unit retrieves the images included in a PDF document, using the `pdfimages` command-line tool. The syntax is:

```
pdfimage -png <target_path> <pdfimages_directory>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a PDF file.

class katana.units.pdf.pdfimages.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['pdf']
PDFs shouldn't come out of this. So no reason to look.

GROUPS = ['pdf', 'pdfimages']
These are "tags" for a unit. Considering it is a pdf unit, "pdf" is included, and the name of this unit "pdfimages".

PRIORITY = 25
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority if this is detected...

RECURSE_SELF = False
Again no PDF from this. So recursion is silly.

evaluate(*case: Any*) → None
Evaluate the target. Run `pdfimages` on the target and recurse on any new found files.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

13.4 katana.units.pdf.pdfinfo — pdfinfo

PDFInfo

This unit checks the PDF information of a given target, using the `pdfinfo` command-line tool. You can optionally pass in arguments, `user_password` and `owner_password` to use with the utility. The syntax is:

```
pdfinfo <target_path> -upw <user_password> -opw <owner_password>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a PDF file.

class katana.units.pdf.pdfinfo.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['pdf']
PDFs shouldn't come out of this. So no reason to look.

DEPENDENCIES = ['pdfinfo']
Required depenencies for this unit "pdfinfo"

GROUPS = ['pdf']
These are "tags" for a unit. Considering it is a pdf unit, "pdf" is included.

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a priority of 60.

RECURSE_SELF = False
Again no PDF from this. So recursion is silly.

evaluate(*case: Any*) → None
Evaluate the target. Run `pdfinfo` on the target and recurse on any new found information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

CHAPTER 14

katana.units.raw — Miscellaneous general operations

These units do small operations on miscellaneous data, tools that could potentially be run across all targets, or fuctionality that might not fit in any other category or unit family.

14.1 katana.units.raw.ascii85 — Decode Ascii85

Decode Ascii85 encoded text

This is done by the Python3 `base64` module which has the `a85decode` function.

class katana.units.raw.ascii85.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.Unit*

GROUPS = ['raw', 'decode', 'ascii85']

These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name "ascii85"

PRIORITY = 60

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a low priority unit, because it is uncommon and highly matching.

evaluate(*case: Any*)

Evaluate the target. Run `base64.a85decode` on the target and recurse on any new found information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

14.2 katana.units.raw.base32 — Decode Base32

Decode Base32 encoded text

This is done by the Python3 `base64` module which has the `b32decode` function.

class katana.units.raw.base32.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.Unit*

GROUPS = ['raw', 'decode', 'base32']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name "base32".

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a low priority.

evaluate(*case*)
Evaluate the target. Run `base64.b32ecode` on the target and recurse on any new found information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

14.3 katana.units.raw.base58 — Decode Base58

Decode Base58 encoded text

This is done by the Python3 `base58` module which has the `b58decode` function.

class katana.units.raw.base58.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode', 'base58']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name "base58".

PATTERN = regex.Regex(b'[a-zA-Z0-9+/]+', flags=regex.A | regex.S | regex.M | regex.V0)

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a low priority.

evaluate(*match*)
Evaluate the target. Run `base58.b58decode` on the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.4 katana.units.raw.base64 — Decode Base64

Decode Base64 encoded text

This is done by the Python3 `base64` module which has the `b64decode` function.

class katana.units.raw.base64.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode', 'base64']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name "base64".

PATTERN = regex.Regex(b'[a-zA-Z0-9+/]{4,}={0,2}', flags=regex.A | regex.S | regex.M |

PRIORITY = 25
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority. Base64 is quick and common and matches fairly unilaterally

evaluate(*match*)
Evaluate the target. Run `base64.b64decode` on the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.5 katana.units.raw.base85 — Decode Base85

Decode Base85 encoded text

This is done by the Python3 `base64` module which has the `b85decode` function.

class katana.units.raw.base85.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode', 'base85']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name "base85".

PATTERN = regex.Regex(b'[\\x21-\\x75]{4,}', flags=regex.A | regex.S | regex.M | regex.

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a low priority, uncommon

evaluate(*match*)
Evaluate the target. Run `base64.b85decode` on the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.6 katana.units.raw.exiftool — Exiftool

Extract metadata with `exiftool`

This unit will extract metadata file using the `exiftool` command-line utility. The syntax runs as:

```
exiftool <target_path>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a file.

class katana.units.raw.exiftool.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*, *keywords=None*)
Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['exiftool']
This unit needs the `exiftool` command-line tool to run.

GROUPS = ['raw', 'file', 'exiftool']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "file", and the name of the unit "exiftool".

PRIORITY = 40
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderate-to-high priority

evaluate(*case*)
Evaluate the target. Run `exiftool` on the target and recurse on any newfound information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

14.7 katana.units.raw.morsecode — Interpret Morsecode

Unit to decode Morsecode

This unit will attempt to read data from Morsecode, both in the International sound mapping as well as the text representation with dots and dashes.

class katana.units.raw.morsecode.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode', 'morsecode']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name "morsecode".

PATTERN = regex.Regex(b'((((dit|dah|di)-?)+)|([.\\-]+))(((((dit|dah|di)-?)+)|([.\\-]+

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderate priority

evaluate(*match*)
Evaluate the target. Translate any morsecode in the target and to its English representation and recurse on any newfound information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.8 katana.units.raw.qrcode — Run zbarimg

Scan QR codes

This unit works with the `pyzbar` module in Python, which is necessary for it to run.

This unit inherits from the *katana.unit.FileUnit* to ensure that the target is in fact an image file.

class katana.units.raw.qrcode.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)
Bases: *katana.unit.FileUnit*

GROUPS = ['raw', 'decode', 'qrcode', 'scan']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", "scan", and the unit name "qrcode".

PRIORITY = 25
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderate priority.

evaluate(*case: Any*)
Evaluate the target. Scan the target with `pyzbar` and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.9 katana.units.raw.strings — Find plaintext strings

Parse plaintext strings from a file with the `strings` command-line tool.

You can supply a minimum length of the data that `strings` will return as an argument `length`. The syntax of the command being run is:

```
strings <target_path> -n <length_argument>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a file.

class katana.units.raw.strings.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*, *keywords=None*)
Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['decode']
THis unit does not recurse to "decode" units, since they are capable of finding their targets within a file by regular expression

DEPENDENCIES = ['strings']
Required depenencies for this unit "strings"

GROUPS = ['raw', 'strings']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, and the name of this unit itself "strings".

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

evaluate(*case: Any*)
Evaluate the target. Run `strings` on the target and recurse on any newfound information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

14.10 katana.units.raw.unbinary — Convert binary to ASCII

Decode data represented as binary values.

This unit will return the data represented in both little-endian notation and in big-endian notation.

class katana.units.raw.unbinary.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the name of the unit itself, "unbinary".

PATTERN = regex.Regex(b'(([01]{7,8}(([01]{7,8})){3,}|[01]{32,}))', flags=regex.A | re
The pattern to match for binary data.

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has the default priority.

evaluate(*match*)
Evaluate the target. Convert the binary data found within the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.11 katana.units.raw.undecimal — Convert decimal to ASCII

Decode data represented as decimal values.

This unit will return the data represented in both little-endian notation and in big-endian notation.

class katana.units.raw.undecimal.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode', 'undecimal']
These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name itself, "undecimal"

PATTERN = regex.Regex(b'[0-9]+(([0-9]+))*', flags=regex.A | regex.V0)
The pattern to match for decimal data.

PRIORITY = 50
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has the default priority.

evaluate(*match*)
Evaluate the target. Convert the decimal data found within the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.12 katana.units.raw.unhexlify — Convert hex to ASCII

Decode data represented as hexadecimal values.

This unit will return the data represented in both little-endian notation and in big-endian notation.

class katana.units.raw.unhexlify.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.RegexUnit*

GROUPS = ['raw', 'decode', 'unhexlify']

These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the unit name itself, "unhexlify".

PATTERN = regex.Regex(b'[0-9a-fA-F]+(([0-9a-fA-F]+))*', flags=regex.A | regex.V0)

The pattern to match for hexadecimal data.

PRIORITY = 50

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderate-high unit priority.

evaluate(*match*)

Evaluate the target. Convert the hexadecimal data found within the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

14.13 katana.units.raw.urldecode — Decode URLs

Decode URL-encoded/percent-ecoded data.

This unit will return the data represented in both little-endian notation and in big-endian notation.

This unit inherits from *katana.unit.PrintableDataUnit* as the targets for this should include data that is often part of a URL.

katana.units.raw.urldecode.**URL_DATA = regex.Regex(b'%[0-9A-Fa-f]{1,2}', flags=regex.A | reg**

The pattern to match for URL encoded data.

class katana.units.raw.urldecode.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.PrintableDataUnit*

GROUPS = ['raw', 'decode', 'urldecode']

These are "tags" for a unit. Considering it is a Raw unit, "raw" is included, as well as the tag "decode", and the name of the unit itself, "urldecode".

PRIORITY = 25

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority.

evaluate(*case: Any*)

Evaluate the target. URL decode the target and recurse on any new found information.

Parameters **match** – A match returned by the `RegexUnit`.

Returns None. This function should not return any data.

CHAPTER 15

katana.units.stego — Steganography

These units handle procedures that are often necessary for challenges in the Steganography category of CTFs.

Note: Often times, these units can take a long amount of time and bottleneck Katana's operations. If you know you do not need these checks, include `--exclude stego` in your command.

15.1 katana.units.stego.audio_spectrogram — Extract Audio Spectrogram

Create an audio spectrogram for audio files

This unit will generate a spectrogram for audio files. It relies heavily on Python libraries such as `pydub` and `pylab`.

This unit inherits from the *katana.unit.FileUnit* to ensure that the target is in fact an audio file.

class katana.units.stego.audio_spectrogram.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

Analyze the audio spectogram of a clip and look for visual text/images

GROUPS = ['audio', 'stego']

These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "audio".

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher than normal priority for matching files

evaluate(*case*)

Evaluate the target. Create an audio spectrogram based off of the given audio file and add it to the results.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.stego.audio_spectrogram.**get_info**(*wav_file: bytes*) → tuple
Get audiodata from the given the file path

15.2 katana.units.stego.dtmf_decode — Decode DTMF Tones

Unit to read values from DTMF tones.

This unit inherits from the *katana.unit.FileUnit* to ensure that the target is in fact an audio file.

..note:

```
Currently, this unit only supports WAVE files (sorry, no MP3s).
```

class katana.units.stego.dtmf_decode.**DTMFdetector**
Bases: object

This is used for the DTMF processing operations. Admittedly, it was found online and adapted to work within Katana.

calc_coeffs()

check(*filename*)

clean_up_processing()

getDTMFfromWAV(*filename*)

goertzel(*sample*)

post_testing()

reset()

class katana.units.stego.dtmf_decode.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.FileUnit*

GROUPS = ['audio', 'stego']
These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "audio".

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority for matching files

evaluate(*case*)
Evaluate the target. Attempt to retrieve the DTMF tones present in the target sound file.

Parameters **case** – A case returned by enumerate. For this unit, the enumerate function is not used.

Returns None. This function should not return any data.

15.3 katana.units.stego.jsteg — Run jsteg

Extract hidden data with jsteg

This unit will extract hidden data file using the jsteg command-line utility. The syntax runs as:

```
jsteg reveal <target_path>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a JPG file.

class katana.units.stego.jsteg.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['jsteg']
Required depenencies for this unit "jsteg"

GROUPS = ['stego', 'image', 'jsteg']
These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "image", and the unit name itself, "jsteg".

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority for matching units

evaluate(*case*)
Evaluate the target. Run `jsteg` on the target and recurse on any newfound information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

15.4 katana.units.stego.snow — Run snow

Extract hidden data with `snow`

This unit will extract hidden data file using the `snow` command-line utility. The syntax runs as:

```
snow <target_path>
```

You can read more about the `snow` tool at the homepage, here: http://www.darkside.com.au/snow/

The unit inherits from *katana.unit.FileUnit* to ensure the target is a file.

class katana.units.stego.snow.**Unit**(*manager: katana.manager.Manager*, *target: katana.target.Target*, *keywords=None*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['snow']
Required depenencies for this unit "snow"

GROUPS = ['stego', 'text', 'snow']
These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "text" and the name of unit itself, "snow".

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority for matching files

evaluate(*case*)
Evaluate the target. Run `snow` on the target and recurse on the standard output.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

15.5 katana.units.stego.steghide — Run steghide

Extract hidden data with `steghide`

This unit will extract hidden data file using the `steghide` command-line utility. First the unit will try with an empty password, and then it will try with the user-supplied password argument. Finally, it will bruteforce with a upplied dictionary file. The syntax runs as:

```
steghide extract -sf <target_path> -p <password> -xf <steghide_directory>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a JPG file.

Note: `steghide` only works on JPG files!

class katana.units.stego.steghide.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['steghide']

Required depenencies for this unit "steghide"

GROUPS = ['stego', 'image']

These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "image".

PRIORITY = 20

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a high priority for matching files

enumerate()

This function will first yield an empty password, then the supplied password argument, then loop through each line of a provided dictionary file. The password will then be used by the `evaluate` function to try and open the encrypted PDF.

evaluate(*password*)

Evaluate the target. Extract any info with steghide and recurse on any new found files.

Parameters **password** – A case returned by `enumerate`. For this unit, `password` will first be an empty password, then the password supplied as an argument, then the contents of a provided dictionary file.

Returns None. This function should not return any data.

15.6 katana.units.stego.stegsolve — Run Stegsolve

Reveal color planes on an image with `stegsolve`.

This unit is a Python implementation of `stegsolve.jar`, which is often used for CTF challenges.

You can supply a `channel` or `plane` index to specifically extract, but if these arguments are not given the unit will bruteforce and grab the least 4 bits of each color channel (R, G, B, typically).

The unit inherits from *katana.unit.FileUnit* to ensure the target is an image file.

class katana.units.stego.stegsolve.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

BLOCKED_GROUPS = ['stego', 'forensics']
Blocked groups.... do not recurse into forensics because running binwalk or foremost on new images serves no real purpose

GROUPS = ['stego', 'image', 'stegsolve']
These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "image", and the name of the unit itself, "stegsolve".

PRIORITY = 70
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a priorty of 70.

RECURSE_SELF = False
Recurssion would be silly in this case.

enumerate()
This function will first yield the `channel` and `plane` that are supplied as arguments by the end-user. If they are not supplied, by default it will loop through all colors channels and the least 4 bits to extract from the target. These `channel` and `plane` pairs will be presented as a tuple, to be used by the `evaluate` function.

evaluate(*case*)
Evaluate the target. Create new images on specific color channels and their specified bit indexes.

Parameters **case** – A case returned by `enumerate`. For this unit, this will be a tuple with the channel (R, G, B) and plane (0-7) to extract.

Returns None. This function should not return any data.

katana.units.stego.stegsolve.**get_plane**(*img*, *data*, *channel: str*, *index: str = 0*)
Get a new image showcasing only one channel and index of an image.

Parameters

- **img** – The Python PIL original image object
- **data** – The pixel data of the original image object
- **channel** – The channel to extract, as a string (e.g. "R", "G", "B")
- **index** – The specific bit index (0-7) you want to extract

Returns A new Python PIL image with only the given channel and index.

15.7 katana.units.stego.whitespace — Check spaces/tabs for binary

Extract hidden data with Whitespace steganography.

This unit will extract hidden data file treating spaces as a binary 0, tabs as a binary 1, and vice versa.

The unit inherits from *katana.unit.FileUnit* to ensure the target is a file.

class katana.units.stego.whitespace.**Unit**(**args*, ***kwargs*)
Bases: *katana.unit.FileUnit*

GROUPS = ['stego', 'whitespace']
These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, and the name of the unit itself, "whitespace".

PRIORITY = 75
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderate priority.

evaluate(*case*)
Evaluate the target. Convert anything that could potentially be whitespace steganography and pass it to Katana.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

`katana.units.stego.whitespace.`**decode_from_whitespace**(*binary_sequence: str*) → str
This is a convenience function to decode a binary sequence.

Parameters **binary_sequence** – A string of 1's and 0's.

Returns The converted data

15.8 `katana.units.stego.zsteg` — Run zsteg

Extract hidden data with `zsteg`

This unit will extract hidden data file using the `zsteg` command-line utility. The syntax runs as:

```
zsteg <arguments> <target_path>
```

This unit will use only preselected arguments to search with `zsteg`. This saves processing time, and still seems to find the majority of flags.

The unit inherits from *`katana.unit.FileUnit`* to ensure the target is a PNG file.

Note: `zsteg` only works with PNG files!

---

class katana.units.stego.zsteg.**Unit**(**args*, ***kwargs*)
Bases: *`katana.unit.FileUnit`*

DEPENDENCIES = ['zsteg']
Depends on the binary "zsteg". This must be in your PATH for this unit to run.

GROUPS = ['stego', 'image', 'zsteg']
These are "tags" for a unit. Considering it is a Stego unit, "stego" is included, as well as the tag "image", and the name of the unit itself, "zsteg".

PRIORITY = 40
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a slightly higher priority of 40.

enumerate() → Generator[Any, None, None]
This will loop through a set of pre-defined arguments for `zsteg` to run with.

Returns Generator of `zsteg` arguments

evaluate(*case: Any*) → None
Evaluate the target. Run `zsteg` on the target and recurse on any newfound information.

Parameters **case** – A case returned by `enumerate`. For this unit, the `case` is an argument to use for `zsteg`.

Returns None. This function should not return any data.

katana.units.stego.zsteg.**permutations = ['b1,rgb,lsb,xy', 'b1,r,lsb,xy', 'b1,rgb,msb,yx',**
This is a pre-defined list of argument to use with zsteg. These options tend to find flags hidden with the LSB steganography technique.

CHAPTER 16

katana.units.tar — TAR File Processing

These units handle procedures that to work with TAR archive files.

Admittedly, this should be bundled to in a larger, "archive" unit package, but this has not yet been done.

16.1 katana.units.tar.extract — Extract TAR archive

TAR archive extraction

This is done with the built-in Python library `tarfile`, so there is

Note that TAR files do not have support for passwords, so that is not implemented here.

The unit inherits from *katana.unit.FileUnit* to ensure the target is a TAR archive.

class katana.units.tar.extract.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

GROUPS = ['tar', 'archive']

The constructor is included just to provide a keyword for the `FileUnit`, ensuring the provided target is in fact a TAR archive.

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = True

In case we have nested TARs, we CAN recurse into ourselves.

evaluate(*case: str*)

Evaluate the target. Extract the target with TAR and recurse on any new found files.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

CHAPTER 17

katana.units.web — Web Application Testing

These units handle procedures that are often necessary for challenges in the Web category of CTFs.

Note: These units are by default *aggressive*: they will automatically perform SQL injections, attempt LFI, bruteforce web pages and more. Ensure that you have *full authorization and permission* to point this at a website.

Admittedly, these should be organized into a framework so that once vulnerabilities are found for a website, they can be shared with sister units and leveraged as needed. This is a large undertaking that is still not completed.

17.1 katana.units.web.basic_img_shell — Upload PHP Shell

Upload a basic PHP web shell to look for a flag file.

This unit will to see if there upload functionality on a webpage, and if it finds one, it will attempt to upload a basic PHP web shell masked inside of a GIF image. That syntax is simply:

```
GIF89a;
<?php system($_GET['c']) ?>
```

If the unit can find the new file that it uploaded, it will attempt to run commands and look for a `flag.txt` or `flag` file on the remote server.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

Warning: This unit automatically attempts to perform malicious actions on the target. **DO NOT** use this in any circumstances where you do not have the authority to operate!

class katana.units.web.basic_img_shell.**Unit**(**args*, ***kwargs*)

Bases: katana.units.web.WebUnit

GROUPS = ['web', 'shell', 'basic_img_shell']
These are "tags" for a unit. Considering it is a web unit, "web" is included, as well as the tag "shell", and the name of the unit itself, "basic_img_shell".

PRIORITY = 60
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat lower priority.

RECURSE_SELF = False
This unit should not recurse on itself.

enumerate()
Yield cases. This function will actually attempt to upload a PHP webshell with a variety of file extensions, like `["php", "gif", "php3", "php5", "php7"]` and yield the proper HTTP action, method, parameters and potentially a file location to reach the uploaded webshell. Running commands takes place within the `evaluate` function.

Returns A generator, yielding a tuple with the found values `(method, action, file, ext, location, file_path)`

evaluate(*case: Any*)
Evaluate the target. Use the uploaded webshell to try and run commands and if command output is shown, find a potential flag location. If a flag file is found, it will attempt to display that flag.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function yields the information necessary to access the newly uploaded webshell.

Returns None. This function should not return any data.

17.2 katana.units.web.basic_nosqli — NoSQL Injection

Basic NoSQL Injection

This will attempt basic NoSQL injection (MongoDB) of the form `"username": {"$gt": ""}, "password": {"$gt": ""},`.

It passes a User-Agent to act as a regular Firefox web browser.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

Warning: This unit automatically attempts to perform malicious actions on the target. **DO NOT** use this in any circumstances where you do not have the authority to operate!

class katana.units.web.basic_nosqli.**Unit**(**args*, ***kwargs*)
Bases: `katana.units.web.WebUnit`

GROUPS = ['web', 'shell', 'basic_nosqli']
These are "tags" for a unit. Considering it is a web unit, "web" is included, as well as the tag "shell", and the name of the unit itself, "basic_nosqli".

PRIORITY = 25
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority.

RECURSE_SELF = False
This unit should not recurse on itself.

evaluate(*case: Any*)
Evaluate the target. Attempt to perform NoSQL injection (MongoDB) on the form found on the target web page.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

17.3 `katana.units.web.basic_sqli` — SQL Injection

Basic SQL Injection

This will attempt basic SQL injection of the form ' OR 1=1 # with varying quotes, comment techniques, and nested SQL clauses.

It passes a User-Agent to act as a regular Firefox web browser.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

Warning: This unit automatically attempts to perform malicious actions on the target. **DO NOT** use this in any circumstances where you do not have the authority to operate!

class katana.units.web.basic_sqli.**Unit**(**args*, ***kwargs*)
Bases: `katana.units.web.WebUnit`

GROUPS = ['web', 'shell', 'basic_sqli']
These are "tags" for a unit. Considering it is a web unit, "web" is included, as well as the tag "shell", and the name of the unit itself, "basic_sqli".

PRIORITY = 25
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority.

RECURSE_SELF = False
This unit should not recurse on itself.

enumerate()
Yield cases. This function will attempt to generate all of the potential payload options for basic SQL injection, between single-quotes versus double-quotes, MySQL-style comments or SQLite-style comments or for delimeters and even nested SQL clauses.

Returns A generator, yielding a tuple with the found values `(method, action, username, password, payload)`

evaluate(*case: Any*)
Evaluate the target. Attempt to perform SQL injection on the form found on the target web page.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function will offer the HTTP method, action, username and password argument names, as well as the changing SQL injection payload to test against the remote server.

Returns None. This function should not return any data.

17.4 katana.units.web.cookies — Check Cookies

View HTTP cookies

This unit will look through all of the different cookies on a website and look for a flag.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

class katana.units.web.cookies.**Unit**(**args*, ***kwargs*)

Bases: katana.units.web.WebUnit

GROUPS = ['web', 'cookies']

These are "tags" for a unit. Considering it is a Web unit, "web" is included, as well as the name of the unit, "cookies".

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = False

This unit should not recurse into itself. It would not make sense to recurse on cookies.

enumerate()

Yield cases. This function will look at the cookies in the requested page and yield each one, to be examined by the `evaluate` function.

Returns A generator, yielding a dictionary with the cookie information (i.e, name=value dictionary).

evaluate(*case*)

Run unit tasks given *case* which was returned from *Unit.enumerate*. This could happen in any thread or process of execution and should be stateless.

17.5 katana.units.web.form_submit — Auto-submit Forms

Basic HTTP form submission

This unit will attempt to submit an HTTP form with no data, if just to find another endpoint accessible on a website.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

class katana.units.web.form_submit.**Unit**(**args*, ***kwargs*)

Bases: katana.units.web.WebUnit

GROUPS = ['web', 'form_submit']

These are "tags" for a unit. Considering it is a Web unit, "web" is included, as well as the name of the unit, "cookies".

PRIORITY = 20

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a higher priority.

RECURSE_SELF = True

This unit should not recurse into itself.

evaluate(*case: Any*)

Evaluate the target. Submit an HTTP form.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

17.6 `katana.units.web.git` — Dump Git Repos

Git Dumper

This unit will detect if a `/.git/` directory is found on a website. If it is, it will pull down all the files and search for flags within the commits and objects inside of the public facing git repository.

This process is threaded, alongside Katana already being threaded... so your mileage may vary.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

Note: This code is shamelessly ripped from https://github.com/arthaud/git-dumper

---

class `katana.units.web.git.`**DownloadWorker**(*pending_tasks*, *tasks_done*, *args*)
Bases: `katana.units.web.git.Worker`

Part of the Git Dumper procedure.

Download a list of files

do_task(*filepath*, *url*, *directory*, *retry*, *timeout*, *unit*, *katana*)

init(*url*, *directory*, *retry*, *timeout*, *unit*, *katana*)

class `katana.units.web.git.`**FindObjectsWorker**(*pending_tasks*, *tasks_done*, *args*)
Bases: `katana.units.web.git.DownloadWorker`

Part of the Git Dumper procedure.

Find objects.

do_task(*obj*, *url*, *directory*, *retry*, *timeout*, *unit*, *katana*)

class `katana.units.web.git.`**FindRefsWorker**(*pending_tasks*, *tasks_done*, *args*)
Bases: `katana.units.web.git.DownloadWorker`

Part of the Git Dumper procedure.

Find refs/

do_task(*filepath*, *url*, *directory*, *retry*, *timeout*, *unit*, *katana*)

class `katana.units.web.git.`**RecursiveDownloadWorker**(*pending_tasks*, *tasks_done*, *args*)
Bases: `katana.units.web.git.DownloadWorker`

Part of the Git Dumper procedure.

Download a directory recursively.

do_task(*filepath*, *url*, *directory*, *retry*, *timeout*, *unit*, *katana*)

class `katana.units.web.git.`**Unit**(**args*, ***kwargs*)
Bases: `katana.units.web.WebUnit`

BAD_MIME_TYPES = ['application/octet-stream']

GROUPS = ['web', 'git']
These are "tags" for a unit. Considering it is a Web unit, "web" is included, as well as the name of the unit, "git".

PRIORITY = 40
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat higher priority.

RECURSE_SELF = False
This unit should not recurse into itself. It would make no sense.

evaluate(*case: Any*)
Evaluate the target. If a `.git` repository is found, download it and look through all of the objects for a flag.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

class katana.units.web.git.**Worker**(*pending_tasks*, *tasks_done*, *args*)
Bases: multiprocessing.context.Process

Part of the Git Dumper procedure.

Worker for process_tasks

do_task(*task*, **args*)

init(**args*)

run()
Method to be run in sub-process; can be overridden in sub-class

katana.units.web.git.**bad_starting_links = [b'#', b'javascript:', b'https://', b'http://',**
This is a blacklist to avoid inline JavaScript, anchors, and external links..

katana.units.web.git.**create_intermediate_dirs**(*path*)
Part of the Git Dumper procedure.

Create intermediate directories, if necessary

katana.units.web.git.**fetch_git**(*unit*, *url*, *directory*, *jobs*, *retry*, *timeout*, *katana*)
Dump a .git repository into the output directory.

This is the core function of the https://github.com/arthaud/git-dumper code.

katana.units.web.git.**get_indexed_files**(*response*)
Part of the Git Dumper procedure.

Return all the files in the directory index webpage.

katana.units.web.git.**get_referenced_sha1**(*obj_file*)
Part of the Git Dumper procedure.

Return all the referenced SHA1 in the given object file

katana.units.web.git.**has_a_bad_start**(*link*)
This is a convenience function just to avoid bad links above

katana.units.web.git.**is_html**(*response*)
Return True if the response is a HTML webpage

katana.units.web.git.**process_tasks**(*initial_tasks*, *worker*, *jobs*, *args=()*, *tasks_done=None*)
Part of the Git Dumper procedure.

Process tasks in parallel.

17.7 katana.units.web.logon_cookies — Check Authentication Cookies

Add or adjust cookies after fake logon.

This unit will attempt to authenticate with the credentials `guest/guest` and then adjust the found cookies to claim that this user has administrator privileges.

It passes a User-Agent to act as a regular Firefox web browser.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

Warning: This unit automatically attempts to perform malicious actions on the target. **DO NOT** use this in any circumstances where you do not have the authority to operate!

class katana.units.web.logon_cookies.**Unit**(**args*, ***kwargs*)

Bases: katana.units.web.WebUnit

GROUPS = ['web', 'cookies', 'logon_cookies']

These are "tags" for a unit. Considering it is a Web unit, "web" is included, as well as the name of the unit, "logon_cookies".

PRIORITY = 30

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has moderately high priority due to speed and broadness of applicability

RECURSE_SELF = False

This unit does not recures into itself. It would not make sense to recurse on cookies

evaluate(*case*)

Evaluate the target. Authenticate to the site with a bogey login and then adjust or add cookies.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

17.8 katana.units.web.robots — Check robots.txt

Check robots.txt

This unit will look through all of the different robots.txt entries on a webpage and look for a flag.

It passes a User-Agent to act as a Google-bot crawler.

This unit inherits from `katana.units.web.WebUnit` as that contains lots of predefined variables that can be used throughout multiple web units.

Warning: This unit automatically attempts to perform malicious actions on the target. **DO NOT** use this in any circumstances where you do not have the authority to operate!

class katana.units.web.robots.**Unit**(**args*, ***kwargs*)
Bases: katana.units.web.WebUnit

GROUPS = ['web', 'robots', 'robots.txt']
These are "tags" for a unit. Considering it is a Web unit, "web" is included, as well as the name of the unit, "robots".

PRIORITY = 30
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat higher priority.

RECURSE_SELF = False
This unit should not recurse into itself. That would be silly.

enumerate()
Yield cases. This function will look at robots.txt page and return each page, to be examined by the evaluate function.

Returns A generator, yielding a string for each URL in robots.txt.

evaluate(*case*)
Evaluate the target. Reach out to every entry in the robots.txt file and look for flags.

Parameters **case** – A case returned by enumerate. For this unit, the enumerate function will yield each URL in the robots.txt file

Returns None. This function should not return any data.

katana.units.web.robots.**headers = {'User-Agent': 'Googlebot/2.1'}**
Include these headers in the unit, to simulate action as the Googlebot crawler.

17.9 katana.units.web.spider — Spider Webpages

Spider web pages

This unit will look through all of the different links on a website and queue each of them as a new target, or link to explore.

This unit inherits from katana.units.web.WebUnit as that contains lots of predefined variables that can be used throughout multiple web units.

Warning: This unit automatically attempts to perform malicious actions on the target. **DO NOT** use this in any circumstances where you do not have the authority to operate!

class katana.units.web.spider.**Unit**(**args*, ***kwargs*)
Bases: katana.units.web.WebUnit

BAD_MIME_TYPES = ['application/octet-stream']
Avoid mime types that are downloadable files.

PRIORITY = 20
Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a somewhat higher priority.

PROTECTED_RECURSE = True
We don't really want to spider on EVERYTHING and start an infinite loop.. We can protect against this once we create a target object and start to "keep track" of links we find in one specific website target

evaluate(*case: Any*)

Evaluate the target. Look for links inside of the target web page and reach out to each of them, queueing them as a new target.

Parameters **case** – A case returned by `enumerate`. For this unit, the `enumerate` function is not used.

Returns None. This function should not return any data.

katana.units.web.spider.**bad_starting_links = [b'#', b'javascript:', b'https://', b'http:/**

Avoid inline JavaScript, anchors, and external links

katana.units.web.spider.**has_a_bad_start**(*link*)

This is a convenience function just to avoid bad links above

CHAPTER 18

katana.units.zip — ZIP File Processing

These units handle procedures that to work with ZIP archive files.

Admittedly, this should be bundled to in a larger, "archive" unit package, but this has not yet been done.

18.1 katana.units.zip.unzip — Unzip/Crack ZIP Password

ZIP file extraction

This unit attempt to extract a ZIP file. First the unit will try with an empty password, and then it will try with the user-supplied password argument. Finally, it will bruteforce with a upplied dictionary file. The process is done with a dependency, using the `unzip` command like so:

```
unzip -P <password> <target_path>
```

The unit inherits from *katana.unit.FileUnit* to ensure the target is a ZIP file.

class katana.units.zip.unzip.**Unit**(**args*, ***kwargs*)

Bases: *katana.unit.FileUnit*

DEPENDENCIES = ['unzip']

This process is done with the `unzip` command because the Python method bottlenecks.

GROUPS = ['zip', 'office', 'archive']

These are "tags" for a unit. Considering it is a zip unit, "zip" is included, as well as a few other key words.

PRIORITY = 25

Priority works with 0 being the highest priority, and 100 being the lowest priority. 50 is the default priorty. This unit has a moderately high priority due to speed and broadness of applicability

RECURSE_SELF = True

In can case we have nested ZIPs, we can recurse into ourselves

enumerate()

This function will first yield an empty password, then the supplied password argument, then loop through

each line of a provided dictionary file. The password will then be used by the `evaluate` function to try and extract the ZIP fike.

evaluate(*case: str*)

Evaluate the target. Extract the target with ZIP and recurse on any new found files.

Parameters **case** – A case returned by `enumerate`. For this unit, `case` will first be an empty password, then the password supplied as an argument, then the contents of a provided dictionary file.

Returns None. This function should not return any data.

CHAPTER 19

Indices and tables

- genindex
- modindex
- search

Python Module Index

k

Index

P

R

S

T

U

V

W