Katnip Documentation

Release 0.2.0

Cisco SAS team

January 26, 2016

Contents

Contents:

CHAPTER 1

Introduction

1.1 What is Katnip?

Katnip is a repository of implementations and extensions for Kitty (link TBD).

While Kitty defines the base classes and syntax, it contains no specific implementation for any of them. So, for example, in order to send a payload over TCP, you need to create some class that extends ServerTarget and is able to send data over TCP, and so on.

Katnip contains such classes. Currently, Katnip contains various implementations of:

- Controllers for servers and clients
- Monitors
- Targets
- Legos
- Templates

1.2 Contribution FAQ

Found a bug? Open an issue.

Have a fix? Great! please submit a pull request.

Want to share you implementation? Thank You! Please submit a pull request.

CHAPTER 2

API Reference

2.1 Katnip API Reference

2.1.1 Subpackages

katnip.controllers package

Subpackages

katnip.controllers.client package Collection of client controllers.

When performing client fuzzing, the main usage of the controller is to trigger the client to start communication with the server stack.

Submodules

katnip.controllers.client.facedancer module This controller is used to signal the SAS version of the Facedancer stack to perform a USB reset.

class katnip.controllers.client.facedancer.**ClientFacedancerController**(*name*, *restart_file='/tmp/restart_facedan*
log-
ger=None)

Bases: kitty.controllers.client.ClientController

ClientFacedancerController is a controller that uses files in /tmp to communicate with the facedancer stack. .. note:: This requires a modified version of the facedancer stack.

RESTART_FILE = '/tmp/restart_facedancer'

__init__(*name*, *restart_file='/tmp/restart_facedancer'*, *logger=None*)

Parameters

- **name** – name of the object
- **controller_port** – the device controller port (i.e. '/dev/ttyACM0')
- **connect_delay** –
- **logger** – logger for the object (default: None)

trigger()
Trigger a data exchange from the tested client

katnip.controllers.client.process module

class katnip.controllers.client.process.**ClientProcessController**(*name*, *process_path*, *process_args*, *process_env=None*, *logger=None*)

Bases: kitty.controllers.client.ClientController

ClientProcessController controls a process by starting it on each trigger. It uses subprocess.Popen and logs the process output (stdout, stderr)

__init__(*name*, *process_path*, *process_args*, *process_env=None*, *logger=None*)

Parameters

- **name** – name of the object
- **process_path** – path to the target executable
- **process_args** – arguments to pass to the process
- **process_env** – the process environment (defualt: None)
- **logger** – logger for this object (default: None)

post_test()
Logs stdout, stderr amd return code of the target process.

teardown()
Stops the process and calls super's teardown.

trigger()
Starts the target in a subprocess

katnip.controllers.client.ssh module

class katnip.controllers.client.ssh.**ClientSshController**(*name*, *username*, *password*, *hostname*, *port*, *command*, *process_name*, *logger=None*)

Bases: kitty.controllers.client.ClientController

ClientSshController controlls a remote process by starting it on each trigger using ssh.

__init__(*name*, *username*, *password*, *hostname*, *port*, *command*, *process_name*, *logger=None*)

Parameters

- **name** – name of the object
- **username** – ssh login username
- **password** – ssh login password
- **hostname** – ssh server ip
- **port** – ssh server port
- **command** – client trigger command
- **process_name** – command process name
- **logger** – logger for this object (default: None)

post_test()
Log output of process, check if crashed

pre_test(*num*)
Creates an SSH connection

teardown()
Closes the SSH connection and calls super's teardown.

trigger()
Trigger the target communication with the server stack.

katnip.controllers.server package

Submodules

katnip.controllers.server.tcp_system module

class katnip.controllers.server.tcp_system.**ServerTcpSystemController**(*name*, *logger*, *proc_name*, *host*, *port*)

Bases: kitty.controllers.base.BaseController

This controller controls a process on a remote machine by sending tcp commands over the network to a local agent on the remote machine to execute the target using popen.

Note: The implementation of the agent is not part of the code it might be added someday, but currently it is not...

__init__(*name*, *logger*, *proc_name*, *host*, *port*)

Parameters

- **name** – name of the object
- **logger** – logger for the object
- **proc_name** – trigger's process name
- **host** – hostname of the agent
- **port** – port of the agent

post_test()

pre_test(*test_number*)

setup()

teardown()

katnip.controllers.server.windbgcontroller module

class katnip.controllers.server.windbgcontroller.**WinAppDbgController**(*name*, *process_path*, *process_args=[]*, *sql_crash_db='sqlite:///crashes.sql*, *logger=None*)

Bases: kitty.controllers.base.BaseController

WinAppDbgController controls a server process by starting it on setup making sure it stays up. It uses winappdbg to attach to the target processes.

__init__(*name*, *process_path*, *process_args=[]*, *sql_crash_db='sqlite:///crashes.sqlite'*, *logger=None*)

Parameters

- **name** – name of the object
- **process_path** – path to the target executable
- **process_args** – arguments to pass to the process
- **attach** – try to attach if process path
- **sql_crash_db** – sql alchemy connection string to crash db (default:sqlite:///crashes.sqlite)
- **logger** – logger for this object (default: None)

post_test()

pre_test(*test_number*)

setup()

Called at the beginning of a fuzzing session. Will start the server up.

teardown()

katnip.legos package

Collection of low level data model structures that can save time when implementing some protocols.

For example, TLV legos to implement templates for TLV-based protocol.

Submodules

katnip.legos.bittorrent module Bittorent file (.torrent) protocol lego. Those legos impelent the bencoding format: https://wiki.theory.org/BitTorrentSpecification#Bencoding

class katnip.legos.bittorrent.**TDict**(*fields={}*, *fuzz_keys=True*, *fuzz_delims=True*, *name=None*)

Bases: kitty.model.low_level.container.Container

Bencoded dictionary. Format: d<bencoded string><bencoded element>e

__init__(*fields={}*, *fuzz_keys=True*, *fuzz_delims=True*, *name=None*)

Parameters

- **fields** – dictionary of strings and torrent fields
- **name** – name of container (default: None)

Fuzz_delims bool (default: True)

class katnip.legos.bittorrent.**TInteger**(*value*, *fuzz_value=True*, *fuzz_delims=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

Bencoded integer. Format: ‘‘ i<integer encoded in base ten ASCII>e‘‘

__init__(*value*, *fuzz_value=True*, *fuzz_delims=True*, *name=None*)

Parameters

- **value** – int, will be enclosed in a Int32
- **name** – name of container (default: None)

Fuzz_value bool (default: True)

Fuzz_delims bool (default: True)

class katnip.legos.bittorrent.**TList**(*fields=[]*, *fuzz_delims=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

Bencoded list. Format: l<bencoded values>e

__init__(*fields=[]*, *fuzz_delims=True*, *name=None*)

Parameters

- **fields** – content of the list, Fields...
- **name** – name of container (default: None)

Fuzz_delims bool (default: True)

class katnip.legos.bittorrent.**TString**(*value*, *fuzz_value=True*, *fuzz_length=True*, *fuzz_delim=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

Bencoded String. Format: <string length encoded in base ten ASCII>:<string data>

__init__(*value*, *fuzz_value=True*, *fuzz_length=True*, *fuzz_delim=True*, *name=None*)

Parameters

- **value** – str, will be enclosed in String
- **fuzz_value** – bool (default: True)
- **fuzz_length** – bool (default: True)
- **fuzz_delim** – bool (default: True)
- **name** – name of container (default: None)

katnip.legos.json module JSON legos - simplified fuzzing of JSON-based protocols

class katnip.legos.json.**JsonArray**(*name*, *values*)
Bases: kitty.model.low_level.container.Container

JSON array field

__init__(*name*, *values*)

Parameters

- **name** – block name
- **values** (list of BaseField) – array members

class katnip.legos.json.**JsonBoolean**(*name*, *value=None*, *fuzzable=True*)
Bases: kitty.model.low_level.container.Container

JSON boolean field

__init__(*name*, *value=None*, *fuzzable=True*)

Parameters

- **name** – block name
- **value** (*bool*) – value to be used, if None - generate both 'true' and 'false' (default: None)
- **fuzzable** – should we fuzz this field (only if value is not None) (default: True)

class katnip.legos.json.**JsonNull**(*name*, *fuzzable=False*)
Bases: kitty.model.low_level.container.Container

JSON Null field

__init__(*name*, *fuzzable=False*)

Parameters

- **name** – block name
- **fuzzable** – should we fuzz this field (default: False)

class katnip.legos.json.**JsonObject**(*name*, *member_dict*, *fuzz_keys=False*)
Bases: kitty.model.low_level.container.Container

JSON object

__init__(*name*, *member_dict*, *fuzz_keys=False*)

Parameters

- **name** – block name
- **member_dict** (dictionary (str, BaseField)) – members of this object
- **fuzz_keys** – should we fuzz the dictionary keys (default: False)

class katnip.legos.json.**JsonString**(*name*, *value*, *fuzzable=True*)
Bases: kitty.model.low_level.container.Container

JSON string field

__init__(*name*, *value*, *fuzzable=True*)

Parameters

- **name** – block name
- **value** – value to be used
- **fuzzable** – should we fuzz this field (default: True)

katnip.legos.json.**dict_to_JsonObject**(*the_dict*, *name=None*, *ctx=None*)
Create a JsonObject from a dictionary. The context parameter is used for recursive calls, no need to pass it from outside.

Parameters

- **the_dict** – dictionary to base the *JsonObject* on
- **ctx** – context for the parser (default: None)

Return type *JsonObject*

Returns JSON object that represents the dictionary

katnip.legos.json.**list_to_JsonArray**(*the_list*, *name=None*, *ctx=None*)
Create a JsonArray from a list. The context parameter is used for recursive calls, no need to pass it from outside.

Parameters

- **the_list** – list to base the JsonArray on
- **ctx** – context for the parser (default: None)

Return type *JsonArray*

Returns JSON object that represents the list

katnip.legos.json.**str_to_json**(*json_str*)
Create a JSON lego based on a json string.

Parameters **json_str** – json string to base the template on

Return type *JsonArray* or *JsonObject*

Returns JSON object or JSON array.

katnip.legos.tlv module TLV (tag/type-length-value) legos. Simplify fuzzing of TLV-based protocol.

class katnip.legos.tlv.**TLV**(*name*, *tag*, *fields=None*, *tag_size=32*, *length_size=32*, *encoder=<kitty.model.low_level.encoder.BitFieldBinEncoder object>*, *fuzzable=True*, *fuzz_tag=False*, *fuzz_length=True*)
Bases: kitty.model.low_level.container.Container

A container for fuzzing TLV elements, it represents a full binary TLV element.

__init__(*name*, *tag*, *fields=None*, *tag_size=32*, *length_size=32*, *encoder=<kitty.model.low_level.encoder.BitFieldBinEncoder object>*, *fuzzable=True*, *fuzz_tag=False*, *fuzz_length=True*)

Parameters

- **name** – name of the tlv element
- **tag** – tag of element
- **fields** – element fields, e.g. value (default: None)
- **tag_size** – size of tag field in bits (default: 32)
- **length_size** – size of length field in bits (default: 32)
- **encoder** – encoder for tag and length fields (default: ENC_INT_BE)
- **fuzzable** – should fuzz the element (default: True)
- **fuzz_tag** – should fuzz the tag value (default: False)
- **fuzz_length** – should fuzz the element length (default: True)

class katnip.legos.tlv.**TLVFactory**(*tag_size=32*, *length_size=32*, *encoder=<kitty.model.low_level.encoder.BitFieldBinEncoder object>*)
Bases: object

Factory class for TLV elements, which allows configuration for all TLV blocks, including:

•Size of the tag/type field in bits

- Size of the length field in bits
- Encoder for tag and length fields

__init__(*tag_size=32*, *length_size=32*, *encoder=<kitty.model.low_level.encoder.BitFieldBinEncoder object>*)

Parameters

- **tag_size** – size of tag field in bits (default: 32)
- **length_size** – size of length field in bits (default: 32)
- **encoder** – encoder for tag and length (default: ENC_INT_BE)

element(*name*, *tag*, *fields=None*, *fuzzable=True*, *fuzz_tag=False*, *fuzz_length=True*)

Generate a TLV element.

Parameters

- **name** – name of the element
- **tag** – value of the element tag
- **fields** – fields of the element may be a field or list of fields - e.g. value (default: None)
- **fuzzable** – should fuzz the element (default: True)
- **fuzz_tag** – should fuzz the tag value (default: False)
- **fuzz_length** – should fuzz the element length (default: True)

katnip.legos.url module URL legos, based on RFC 1738 and others this module containes a basic URL lego, as well as the following specific scheme: HTTP, HTTPS, FTP, FTPS

Todo

URL fragments

---

class katnip.legos.url.**DecimalNumber**(*value*, *num_bits=64*, *signed=False*, *fuzzable=True*, *name=None*)

Bases: kitty.model.low_level.container.OneOf

Decimal number fuzzing. It's main strategy is to fuzz both values (integer vulnerabilities) and format (string vulnerabilities).

__init__(*value*, *num_bits=64*, *signed=False*, *fuzzable=True*, *name=None*)

Parameters

- **value** – default integer value
- **num_bits** – number of bit in the integer (default: 64)
- **signed** – can the value be negative (default: False)
- **fuzzable** – should fuzz the container (default: True)
- **name** – name of container (default: None)

class katnip.legos.url.**EmailAddress**(*username*, *hostname*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Bases: kitty.model.low_level.container.Container

Container to fuzz email address

__init__(*username*, *hostname*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Parameters

- **username** – email username
- **hostname** – email hostname
- **fuzz_delim** – should fuzz the delimiter (default: True)
- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

classmethod from_string(*the_str*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

class katnip.legos.url.**EmailUrl**(*email*, *scheme='mailto'*, *fuzz_scheme=True*, *fuzz_user=True*, *fuzz_host=True*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Bases: *katnip.legos.url.Url*

__init__(*email*, *scheme='mailto'*, *fuzz_scheme=True*, *fuzz_user=True*, *fuzz_host=True*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Parameters

- **email** (*EmailAddress*) – the email address
- **scheme** – URL scheme (default: 'mailto')
- **fuzz_scheme** – should fuzz the URL scheme (default: True)
- **fuzz_user** – should fuzz the username (default: True)
- **fuzz_host** – should fuzz the host (default: True)
- **fuzz_delim** – should fuzz the delimiter (default: True)
- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

classmethod from_string(*the_url*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

class katnip.legos.url.**FType**(*the_type*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Bases: kitty.model.low_level.container.Container

Container to fuzz the FTP Type of FTP URL

```
ftptype         = "A" | "I" | "D" | "a" | "i" | "d"
```

__init__(*the_type*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Parameters

- **the_type** (*str*) – the FTP type
- **fuzz_delims** – should fuzz delimiters (default: True)
- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

class katnip.legos.url.**FtpUrl**(*scheme='ftp'*, *login=None*, *hostport=None*, *path=None*, *ftype=None*, *fuzz_scheme=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Bases: *katnip.legos.url.Url*

Container to fuzz FTP URLs

```
    ftpurl          = "ftp://" login [ "/" fpath [ ";type=" ftptype ]]
    fpath           = fsegment *[ "/" fsegment ]
    fsegment        = *[ uchar | "?" | ":" | "@" | "&" | "=" ]
    ftptype         -> see FType
```

__init__(*scheme='ftp'*, *login=None*, *hostport=None*, *path=None*, *ftype=None*, *fuzz_scheme=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Parameters

- **scheme** – URL scheme name (default: 'ftp')
- **login** (instance of `BaseField` recommend using *Login*) – the login information (default: None)
- **hostport** (*katnip.legos.url.HostPort* object (default: None)) – FTP host and port
- **path** (*katnip.legos.url.Path* object (default: None)) – file path
- **ftype** (*katnip.legos.url.FType* object (default: None)) – FTP type
- **fuzz_scheme** – should fuzz the URL scheme (default: True)
- **fuzz_delims** – should fuzz the delimiters (default: True)
- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

classmethod from_string(*the_url*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Create an FtpUrl Lego from string

Parameters

- **the_url** – the url string
- **fuzz_delims** – should fuzz delimiters (default: True)
- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

class katnip.legos.url.**HostName**(*host=''*, *fuzz_delims=False*, *fuzzable=True*, *name=None*)

Bases: kitty.model.low_level.container.Container

Container to fuzz the hostname

```
    host            = hostname | hostnumber
    hostname        = *[ domainlabel "." ] toplabel
```

__init__(*host=''*, *fuzz_delims=False*, *fuzzable=True*, *name=None*)

Parameters

- **host** (*str*) – hostname (default: '')
- **fuzz_delims** – should fuzz the delimiters (default: False)
- **fuzzable** – should fuzz the container (default: True)
- **name** – name of container (default: None)

class katnip.legos.url.**HostPort**(*host*, *port=None*, *fuzz_host=True*, *fuzz_port=True*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Bases: kitty.model.low_level.container.Container

Container for fuzzing the host/port of the URL.

```
hostport        = host [ ":" port ]
port            = digits
```

__init__(*host*, *port=None*, *fuzz_host=True*, *fuzz_port=True*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Parameters

- **host** (str or instance of `BaseField` recommend using *HostName*) – hostname
- **port** – port number (default: None)
- **fuzz_host** – should fuzz the hostname (default: True)
- **fuzz_port** – should fuzz the port (default: True)
- **fuzz_delim** – should fuzz the delimiter (default: True)
- **fuzzable** – should fuzz the container (default: True)
- **name** – name of container (default: None)

class `katnip.legos.url.`**HttpUrl**(*scheme='http'*, *login=None*, *hostport=None*, *path=None*, *search=None*, *fuzz_scheme=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Bases: *katnip.legos.url.Url*

Container to fuzz Http(s) URL

```
httpurl         = "http://" hostport [ "/" hpath [ "?" search ]]
hpath           = hsegment *[ "/" hsegment ]
hsegment        = *[ uchar | ";" | ":" | "@" | "&" | "=" ]
search          = *[ uchar | ";" | ":" | "@" | "&" | "=" ]
```

__init__(*scheme='http'*, *login=None*, *hostport=None*, *path=None*, *search=None*, *fuzz_scheme=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Parameters

- **scheme** – URL scheme name (default: 'http')
- **login** (instance of `BaseField` recommend using *Login*) – the login information (default: None)
- **hostport** – *katnip.legos.url.HostPort* object, must be set (default: None)
- **path** – Path object (default: None)
- **search** – Search object (default: None)
- **fuzz_scheme** – should fuzz the URL scheme (default: True)
- **fuzz_delims** – should fuzz the delimiters (default: True)
- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

classmethod from_string(*the_url*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Create an HttpUrl Lego from string

Parameters

- **the_url** – the url string
- **fuzz_delims** – should fuzz delimiters (default: True)

- **fuzzable** – is the container fuzzable (default: True)
- **name** – name of the container (default: None)

class katnip.legos.url.**IpUrl**(*scheme*, *login*, *url_path=None*, *fuzz_scheme=True*, *fuzz_login=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Bases: *katnip.legos.url.Url*

IP-based URL

```
ip-schemepart  = "//" login [ "/" urlpath ]
login -> see Login class
alphadigit     = alpha | digit
hostnumber     = digits "." digits "." digits "." digits
user           = *[ uchar | ";" | "?" | "&" | "=" ]
password       = *[ uchar | ";" | "?" | "&" | "=" ]
urlpath        = *xchar    ; depends on protocol see section 3.1
```

__init__(*scheme*, *login*, *url_path=None*, *fuzz_scheme=True*, *fuzz_login=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Parameters

- **scheme** (str or instance of BaseField) – url scheme
- **login** (str or instance of BaseField recommend using *Login*) – the login information
- **url_path** – the url path (default: None)
- **fuzz_scheme** – should fuzz scheme (default: True)
- **fuzz_login** – should fuzz login (default: True)
- **fuzz_delims** – should fuzz delimiters (default: True)
- **fuzzable** – should fuzz the container (default: True)
- **name** – name of container (default: None)

class katnip.legos.url.**Login**(*username=None*, *password=None*, *fuzz_username=True*, *fuzz_password=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Bases: kitty.model.low_level.container.Container

Container to fuzz the login part of the URL

```
login          = [ user [ ":" password ] "@" ]
```

__init__(*username=None*, *password=None*, *fuzz_username=True*, *fuzz_password=True*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Parameters

- **username** – user name (default: None)
- **password** – password (default: None)
- **fuzz_username** – should fuzz username (default: True)
- **fuzz_password** – should fuzz password (default: True)
- **fuzz_delims** – should fuzz delimiters (default: True)
- **fuzzable** – should fuzz the container (default: True)
- **name** – name of container (default: None)

class katnip.legos.url.**Path**(*path=None*, *path_delim='/'*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

Container to fuzz the path of the URL

__init__(*path=None*, *path_delim='/'*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)

Parameters

- **path** (*str*) – path string
- **path_delim** – delimiter in the path str
- **fuzz_delims** – should fuzz the delimiters (default: False)
- **name** – name of container (default: None)
- **fuzzable** – should fuzz the container (default: True)

class katnip.legos.url.**Search**(*search=''*, *fuzz_delims=False*, *fuzzable=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

Container to fuzz the search part of the URL

Todo

real implementation (parse search string etc.)

__init__(*search=''*, *fuzz_delims=False*, *fuzzable=True*, *name=None*)

Parameters

- **search** – search string (default: '')
- **fuzz_delims** – should fuzz the delimiters (default: False)
- **name** – name of container (default: None)
- **fuzzable** – should fuzz the container (default: True)

class katnip.legos.url.**Url**(*scheme*, *parts*, *fuzz_scheme=True*, *fuzz_parts=True*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

Base container for fuzzing URLs.

```
genericurl = scheme ":" schemepart
```

__init__(*scheme*, *parts*, *fuzz_scheme=True*, *fuzz_parts=True*, *fuzz_delim=True*, *fuzzable=True*, *name=None*)

Parameters

- **scheme** (str or instance of BaseField) – url scheme
- **parts** (str or instance of BaseField) – url parts (i.e. content)
- **fuzz_scheme** – should fuzz scheme (default: True)
- **fuzz_parts** – should fuzz parts (default: True)
- **fuzz_delim** – should fuzz delimiters (default: True)
- **fuzzable** – should fuzz the container (default: True)
- **name** – name of container (default: None)

katnip.legos.url.**url_from_string**(*url*, *fuzz_delims=True*, *fuzzable=True*, *name=None*)
Create a URL from string, only URLs with supported schemes will result in a lego. In the rest of the cases, an exception will be raised.

Parameters

- **url** – the URL string
- **fuzz_delims** – should fuzz delimiters (default: True)
- **fuzzable** – should the resulted container be fuzzable (default: True)
- **name** – name of the resulted container (default: None)

katnip.legos.xml module XML (tag/type-length-value) legos. Simplify template creation of XML-based protocol.

class katnip.legos.xml.**XmlAttribute**(*name*, *attribute*, *value*, *fuzz_attribute=False*, *fuzz_value=True*)
Bases: kitty.model.low_level.container.Container

XML attribute field, consists of tag and value

__init__(*name*, *attribute*, *value*, *fuzz_attribute=False*, *fuzz_value=True*)

Parameters

- **name** – name of the block
- **attribute** – attribute
- **value** (*str/unicode/int*) – value of the attribute
- **fuzz_attribute** – should we fuzz the attribute field (default: False)
- **fuzz_value** – should we fuzz the value field (default: True)

class katnip.legos.xml.**XmlElement**(*name*, *element_name*, *attributes=[]*, *content=None*, *fuzz_name=True*, *fuzz_content=False*, *delimiter=''*)
Bases: kitty.model.low_level.container.Container

XML element field

__init__(*name*, *element_name*, *attributes=[]*, *content=None*, *fuzz_name=True*, *fuzz_content=False*, *delimiter=''*)

Parameters

- **name** – name of the field
- **element_name** – element name
- **attributes** (*list*) – list of attributes of this element (default: [])
- **content** (*str/unicode/int/[XmlElement]*) – content of this element (default=None)
- **fuzz_name** – should we fuzz the element name
- **fuzz_content** – should we fuzz the content (n/a for XmlElement)

katnip.monitors package

Submodules

katnip.monitors.network module Network monitor

class katnip.monitors.network.**NetworkMonitor**(*interface*, *dir_path*, *name*, *logger=None*)
Bases: kitty.monitors.base.BaseMonitor

NetworkMonitor is a monitor for network activity on a specific interface. It runs on a separate thread, and currently requires root permissions or CAP_NET_RAW capabilities on Linux.

__init__(*interface*, *dir_path*, *name*, *logger=None*)

Parameters

- **interface** – name of interface to listen to
- **dir_path** – path to store captured pcaps
- **name** – name of the monitor
- **logger** – logger for the monitor instance

post_test()
Store the pcap.

pre_test(*test_number*)
Clean the packet list.

setup()
Open the L2socket.

teardown()
Close the L2socket.

katnip.monitors.serial module

class katnip.monitors.serial.**SerialMonitor**(*name*, *dev_name=None*, *baudrate=115200*, *capture_dir='.'*, *logger=None*)
Bases: kitty.monitors.base.BaseMonitor

SerialMonitor monitors the output of a serial connection by looking for a pattern in the serial output.

This monitor captures all the received data from the serial, but it is also able to detect successful/failed tests by looking for specific patterns in the serial output.

Note: The monitor can work either with a success pattern (failure if pattern was not found) or with a failure pattern (success if pattern was not found)

__init__(*name*, *dev_name=None*, *baudrate=115200*, *capture_dir='.'*, *logger=None*)

Parameters

- **name** – name of the monitor object
- **dev_name** – serial device
- **baudrate** – serial baudrate
- **capture_dir** – where to store the captured serial output
- **logger** – logger for the monitor object

post_test()

pre_test(*test_number*)

set_failure_pattern(*failure_pattern*)
Set a pattern that declares the test as failed if received

Parameters **failure_pattern** (*str*) – regular expression pattern of output that signifies failure (e.g. potential bug there)

set_success_pattern(*success_pattern*)
Set a pattern that declares the test successful if received

Parameters **success_pattern** (*str*) – regular expression pattern of output that signifies success (e.g. no bug there)

setup()

teardown()

katnip.monitors.ssh module

class katnip.monitors.ssh.**SSHMonitor**(*name*, *username*, *password*, *hostname*, *port*, *status_command*, *restart_command=None*, *logger=None*)
Bases: kitty.monitors.base.BaseMonitor

SSHMonitor monitors target ip and runs a command over SSH in case it is not responding.

__init__(*name*, *username*, *password*, *hostname*, *port*, *status_command*, *restart_command=None*, *logger=None*)

Parameters

- **name** – name of the object
- **username** – ssh login username
- **password** – ssh login password
- **hostname** – ssh server ip
- **port** – ssh server port
- **status_command** – command to make sure target is alive
- **restart_command** – command to restart the target in case it is deadore
- **logger** – logger for this object (default: None)

post_test()

pre_test(*test_number*)

setup()

teardown()

katnip.monitors.ssh_file module

class katnip.monitors.ssh_file.**SshFileMonitor**(*name*, *username*, *password*, *hostname*, *port*, *file_mask*, *local_dir*, *fail_if_exists=True*, *setup_commands=[]*, *on_fail_command=None*, *on_fail_delay=0*, *logger=None*)
Bases: kitty.monitors.base.BaseMonitor

SshFileMonitor monitors for files using a file_mask. If found - moves files to local folder, renaming with test number.

X_pre_test(*test_number*)

__init__(*name*, *username*, *password*, *hostname*, *port*, *file_mask*, *local_dir*, *fail_if_exists=True*, *setup_commands=[]*, *on_fail_command=None*, *on_fail_delay=0*, *logger=None*)

Parameters

- **name** – name of the object
- **username** – ssh login username
- **password** – ssh login password
- **hostname** – ssh server ip
- **port** – ssh server port
- **file_mask** – file_mask to fetch
- **local_dir** – local_path to store fetched files
- **fail_if_exists** – fail test if file exists (default: True)
- **logger** – logger for this object (default: None)

post_test()

setup()
Called at the begining of the fuzzing session

teardown()

katnip.monitors.telnet module TelnetMonitor monitors the output of a telnet connection by looking for a pattern in the command output

class katnip.monitors.telnet.**TelnetMonitor**(*name*, *username*, *password*, *host*, *port=23*, *cmd_timeout=3*, *capture_dir='.'*, *logger=None*)
Bases: kitty.monitors.base.BaseMonitor

__init__(*name*, *username*, *password*, *host*, *port=23*, *cmd_timeout=3*, *capture_dir='.'*, *logger=None*)

Parameters

- **name** – name of the monitor
- **username** – remote username
- **password** – remote password
- **host** – telnet host
- **port** – telnet port (default: 23)
- **cmd_timeout** – timtout for running the command (default: 3)
- **capture_dir** – where to store the telnet output (default: ='.')
- **logger** – logger for the object (default: None)

add_monitor_cmd(*cmd*, *expected_output=None*)

add_post_test_cmd(*cmd*, *expected_output=None*)

add_pre_test_cmd(*cmd*, *expected_output=None*)

post_test()

pre_test(*test_number*)

set_failure_pattern(*failure_pattern*)
set a pattern that declares the test a failure if received

set_monitor_command(*cmd*)

set_success_pattern(*success_pattern*)
set a pattern that declares the test successful if received

setup()

teardown()

katnip.targets package

Submodules

katnip.targets.file module

class katnip.targets.file.**FileTarget**(*name*, *file_path*, *base_name*, *postfix=None*, *logger=None*)
Bases: kitty.targets.server.ServerTarget

FileTarget will create files with the fuzzed payloads

__init__(*name*, *file_path*, *base_name*, *postfix=None*, *logger=None*)

Parameters

- **name** – name of the target
- **file_path** – path to stores files at
- **base_name** – base file name, it will be appended by the test number
- **postfix** – filename postfix (default: None)
- **logger** – logger for the object (default: None)

Example

```
FileTarget('FileTarget', '/tmp', 'fuzzed', '.bin')
```

Will generate the followinf files:

```
/tmp/fuzzed_0.bin
/tmp/fuzzed_1.bin
/tmp/fuzzed_2.bin
...
```

pre_test(*test_num*)

katnip.targets.raw_udp module

class katnip.targets.raw_udp.**RawUdpTarget**(*name*, *interface*, *host*, *port*, *timeout=None*, *logger=None*)
Bases: *katnip.targets.udp.UdpTarget*

RawUdpTarget is implementation of a UDP target using a raw socket

__init__(*name*, *interface*, *host*, *port*, *timeout=None*, *logger=None*)

Parameters

- **name** – name of the target
- **interface** – interface name
- **host** – host ip (to send data to) currently unused
- **port** – port to send to

- **timeout** – socket timeout (default: None)
- **logger** – logger for the object (default: None)

katnip.targets.ssl module

class katnip.targets.ssl.**SslTarget**(*name*, *host*, *port*, *timeout=None*, *logger=None*)

Bases: *katnip.targets.tcp.TcpTarget*

SslTarget is an implementation of SSL target, used for testing HTTPs etc.

__init__(*name*, *host*, *port*, *timeout=None*, *logger=None*)

Parameters

- **name** – name of the target
- **host** – host ip (to send data to) currently unused
- **port** – port to send to
- **timeout** – socket timeout (default: None)
- **logger** – logger for the object (default: None)

katnip.targets.tcp module

class katnip.targets.tcp.**TcpTarget**(*name*, *host*, *port*, *max_retries=10*, *timeout=None*, *logger=None*)

Bases: kitty.targets.server.ServerTarget

TcpTarget is implementation of a TCP target for the ServerFuzzer

__init__(*name*, *host*, *port*, *max_retries=10*, *timeout=None*, *logger=None*)

Parameters

- **name** – name of the target
- **host** – host ip (to send data to) currently unused
- **port** – port to send to
- **max_retries** – maximum connection retries (default: 10)
- **timeout** – socket timeout (default: None)
- **logger** – logger for the object (default: None)

post_test(*test_num*)

Called after a test is completed, perform cleanup etc.

pre_test(*test_num*)

katnip.targets.udp module

class katnip.targets.udp.**UdpTarget**(*name*, *host*, *port*, *timeout=None*, *logger=None*)

Bases: kitty.targets.server.ServerTarget

UdpTarget is implementation of a UDP target

__init__(*name*, *host*, *port*, *timeout=None*, *logger=None*)

Parameters

- **name** – name of the target
- **host** – host ip (to send data to) currently unused

- **port** – port to send to
- **timeout** – socket timeout (default: None)
- **logger** – logger for the object (default: None)

post_test(*test_num*)

pre_test(*test_num*)

set_binding(*host*, *port*, *expect_response=False*)
enable binding of socket to given ip/address

katnip.templates package

Templates for different protocols

Note: the templates themselves are not documented in here, please take a look at the source for the list of templates.

Submodules

katnip.templates.bittorent module Tempalte of a bittorent file.

Use it directly, or copy and modify it, as it generates many (> 1M) payloads.

This template is based on the MetaInfo file structure: https://wiki.theory.org/BitTorrentSpecification#Metainfo_File_Structure

katnip.templates.bootp module bootp basic Template

katnip.templates.ftp module FTP Protocol command templates. Based on RFC 959 (https://www.ietf.org/rfc/rfc959.txt)

Nice presentation about FTP can be found here: http://www.csun.edu/~jeffw/Semesters/2006Fall/COMP429/Presentations/Ch25-FTP.pdf

class katnip.templates.ftp.**TelnetString**(*command*, *optional=False*, *parameter=None*, *name=None*)
Bases: kitty.model.low_level.container.Template

represents: [Command]<SP>[Parameter]<CRLF>

__init__(*command*, *optional=False*, *parameter=None*, *name=None*)

Parameters

- **command** – command string
- **optional** – has optional parameter (default: False)
- **parameter** – optional parameter string(default: None)
- **name** – name of the field (default: None)

katnip.templates.png module PNG Templates - There's still work to be done

class katnip.templates.png.**Chunk**(*chunk_type*, *data_fields=None*, *fuzzable=True*, *name=None*)
Bases: kitty.model.low_level.container.Container

PNG Chunk

__init__(*chunk_type*, *data_fields=None*, *fuzzable=True*, *name=None*)

Parameters

- **chunk_type** – four-char string (e.g. IHDR, iTXt, etc.)
- **data_fields** (*field or list of fields*) – chunk data(default: None)
- **fuzzable** – is the field fuzzable(default: True)
- **name** – name of the field(default: None)

katnip.templates.png.**compression_func**(*s*)

class katnip.templates.png.**iTXt**(*keyword*, *data*, *fuzzable=True*, *name='iTXt'*, *compressed=False*)
Bases: *katnip.templates.png.Chunk*

iTXt chunk.

__init__(*keyword*, *data*, *fuzzable=True*, *name='iTXt'*, *compressed=False*)

Parameters

- **keyword** – chunk keyword
- **data** (*str*) – chunk data
- **fuzzable** – is the field fuzzable (default: True)
- **name** – name of the field (default: 'tEXt')
- **compressed** – is data compressed (default: False)

class katnip.templates.png.**tEXt**(*keyword*, *data*, *fuzzable=True*, *name='tEXt'*)
Bases: *katnip.templates.png.Chunk*

tEXt chunk.

__init__(*keyword*, *data*, *fuzzable=True*, *name='tEXt'*)

Parameters

- **keyword** – chunk keyword
- **data** (*str*) – chunk data
- **fuzzable** – is the field fuzzable (default: True)
- **name** – name of the field (default: 'tEXt')

class katnip.templates.png.**zTXt**(*keyword*, *data*, *fuzzable=True*, *name='zTXt'*)
Bases: *katnip.templates.png.Chunk*

zTXt chunk.

__init__(*keyword*, *data*, *fuzzable=True*, *name='zTXt'*)

Parameters

- **keyword** – chunk keyword
- **data** (*str*) – chunk data

- **fuzzable** – is the field fuzzable (default: True)
- **name** – name of the field (default: 'zTXt')

katnip.templates.usb module USB Protocol tempaltes. The templates here are based on the USB 2.0 spec. All page / section references are for the USB 2.0 spec document The USB 2.0 may be downloaded from: http://www.usb.org/developers/docs/usb20_docs/usb_20_042814.zip

class katnip.templates.usb.**Descriptor**(*name*, *fields*)
Bases: kitty.model.low_level.container.Template

USB descriptor template.

__init__(*name*, *fields*)

class katnip.templates.usb.**SizedPt**(*name*, *fields*)
Bases: kitty.model.low_level.container.Container

Sized part of a descriptor. It receives all fields excepts of the size field and adds it.

__init__(*name*, *fields*)

Parameters

- **name** – name of the Container
- **fields** – list of fields in the container

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

k

Symbols

A

C

D

E

F

H

I

J

K

L

N

P

R

S

T

U

W

X

Z