WAF-A-MoLE

Oct 24, 2022

Contents:

A *guided mutation-based fuzzer* for ML-based Web Application Firewalls, inspired by AFL and based on the FuzzingBook by Andreas Zeller et al.

Given an input SQL injection query, it tries to produce a *semantic invariant* query that is able to bypass the target WAF. You can use this tool for assessing the robustness of your product by letting WAF-A-MoLE explore the solution space to find dangerous "blind spots" left uncovered by the target classifier.

CHAPTER 1

Architecture

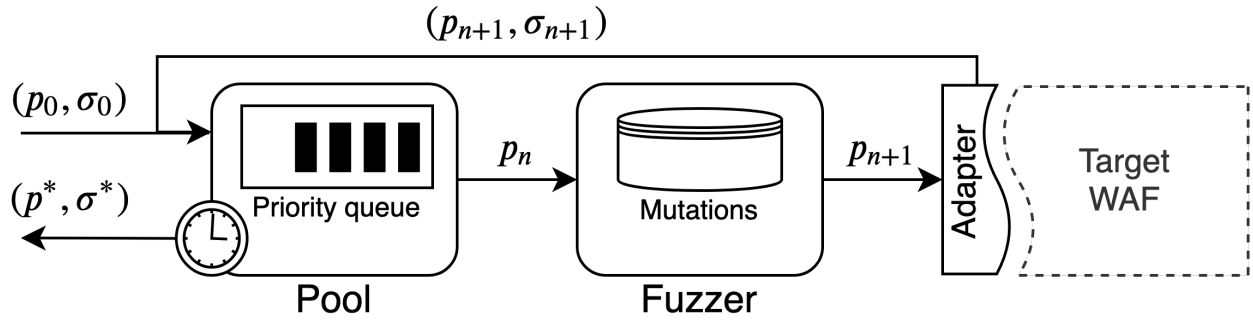

Fig. 1: WAF-A-MoLE Architecture

WAF-A-MoLE takes an initial payload and inserts it in the payload **Pool**, which manages a priority queue ordered by the WAF confidence score over each payload.

During each iteration, the head of the payload Pool is passed to the **Fuzzer**, where it gets randomly mutated, by applying one of the available mutation operators.

1.1 Mutation operators

Mutations operators are all *semantics-preserving* and they leverage the high expressive power of the SQL language (in this version, MySQL).

Below are the mutation operators available in the current version of WAF-A-MoLE.

Mutation	Example
Case Swapping	`admin' OR 1=1# admin' oR 1=1#`
Whitespace Substitution	`admin' OR 1=1# admin'\t\rOR\n1=1#`
Comment Injection	`admin' OR 1=1# admin'/**/OR 1=1#`
Comment Rewriting	`admin'/**/OR 1=1# admin'/*xyz*/OR 1=1#abc`
Integer Encoding	`admin' OR 1=1# admin' OR 0x1=(SELECT 1)#`
Operator Swapping	`admin' OR 1=1# admin' OR 1 LIKE 1#`
Logical Invariant	`admin' OR 1=1# admin' OR 1=1 AND 0<1#`

1.2 Components

1.2.1 wafamole

1.2.1.1 wafamole package

1.2.1.1.1 Subpackages

1.2.1.1.1.1 wafamole.evasion package

1.2.1.1.1.2 Submodules

1.2.1.1.1.3 wafamole.evasion.engine module

1.2.1.1.1.4 wafamole.evasion.evasion module

1.2.1.1.1.5 wafamole.evasion.random module

1.2.1.1.1.6 Module contents

1.2.1.1.1.7 wafamole.models package

1.2.1.1.1.8 Submodules

1.2.1.1.1.9 wafamole.models.keras_model module

1.2.1.1.1.10 wafamole.models.model module

Abstract machine learning model.

class wafamole.models.model.**Model**

Bases: object

Abstract machine learning model wrapper.

classify(*value: object*)

It returns the probability of belonging to a particular class. It calls the extract_features function on the input value to produce a feature vector.

Parameters **value** (*object*) – Input value

Returns the confidence of the malicious class.

Return type float

extract_features(*value: object*)
It extract a feature vector from the input object.

Parameters **value** (*object*) – An input point that belongs to the input space of the wrapped model.

Returns array containing the feature vector of the input value.

Return type feature_vector (numpy ndarray)

Raises NotImplementedError – this method needs to be implemented

1.2.1.1.1.11 wafamole.models.sklearn_model module

1.2.1.1.1.12 Module contents

1.2.1.1.1.13 wafamole.payloadfuzzer package

1.2.1.1.1.14 Submodules

1.2.1.1.1.15 wafamole.payloadfuzzer.fuzz_utils module

wafamole.payloadfuzzer.fuzz_utils.**filter_candidates**(*symbols*, *payload*)
It removes all the symbols that are not contained inside the input payload string.

Parameters

- **symbols** (*dict*) – dictionary of symbols to filter (using the key)
- **payload** (*str*) – the payload to use for the filtering

Raises TypeError – bad types passed as argument

Returns a list containing all the symbols that are contained inside the payload.

Return type list

wafamole.payloadfuzzer.fuzz_utils.**num_contradiction**()
Returns a random contradiction explicit using numbers chosen from a fixed set.

Returns string containing a contradiction

Return type (str)

wafamole.payloadfuzzer.fuzz_utils.**num_tautology**()
Returns a random tautology explicit using numbers chosen from a fixed set.

Returns string containing a tautology

Return type (str)

wafamole.payloadfuzzer.fuzz_utils.**random_char**(*spaces=True*)
Returns a random character.

Keyword Arguments **spaces** (*bool*) – include spaces [default = True]

Raises TypeError – spaces not bool

Returns random character

Return type str

wafamole.payloadfuzzer.fuzz_utils.**random_string**(*max_len=5*, *spaces=True*)
It creates a random string.

Keyword Arguments

- **max_length** (*int*) – the maximum length of the string [default=5]
- **spaces** (*bool*) – if True, all the printable character will be considered. Else, only letters and digits [default=True]

Raises TypeError – bad type passed as argument

Returns random string

Return type (str)

wafamole.payloadfuzzer.fuzz_utils.**replace_nth**(*candidate*, *sub*, *wanted*, *n*)
Replace the n-th occurrence of a portion of the candidate with wanted.

Parameters

- **candidate** (*str*) – the string to be modified
- **sub** (*str*) – regexp containing what to substitute
- **wanted** (*str*) – the string that will replace sub
- **n** (*int*) – the index of the occurrence to replace

Raises TypeError – bad type passed as arguments

Returns the modified string

Return type (str)

wafamole.payloadfuzzer.fuzz_utils.**replace_random**(*candidate*, *sub*, *wanted*)
Replace one picked at random of the occurrence of sub inside candidate with wanted.

Parameters

- **candidate** (*str*) – the string to be modified
- **sub** (*str*) – regexp containing what to substitute
- **wanted** (*str*) – the string that will replace sub

Raises TypeError – bad type passed as arguments

Returns the modified string

Return type (str)

wafamole.payloadfuzzer.fuzz_utils.**string_contradiction**()
Returns a random contradiction chosen from a fixed set.

Returns string containing a contradiction

Return type (str)

wafamole.payloadfuzzer.fuzz_utils.**string_tautology**()
Returns a random tautology chosen from a fixed set.

Returns string containing a tautology

Return type (str)

1.2.1.1.1.16 wafamole.payloadfuzzer.sqlfuzzer module

Strategies and fuzzer class module

class wafamole.payloadfuzzer.sqlfuzzer.**SqlFuzzer**(*payload*)
Bases: object

SqlFuzzer class

current()

fuzz()

reset()

strategies = [<function spaces_to_comments>, <function random_case>, <function swap_ke

wafamole.payloadfuzzer.sqlfuzzer.**change_tautologies**(*payload*)

wafamole.payloadfuzzer.sqlfuzzer.**comment_rewriting**(*payload*)

wafamole.payloadfuzzer.sqlfuzzer.**logical_invariant**(*payload*)
Adds an invariant boolean condition to the payload

E.g., something OR False

Parameters **payload** –

wafamole.payloadfuzzer.sqlfuzzer.**random_case**(*payload*)

wafamole.payloadfuzzer.sqlfuzzer.**reset_inline_comments**(*payload: str*)
Remove randomly chosen multi-line comment content. :param payload: query payload string

Returns payload modified

Return type str

wafamole.payloadfuzzer.sqlfuzzer.**shuffle_integers**(*payload*)
Replace number=number or number LIKE number cases with a digit + letter combination of the number's size

e.g. SELECT admins FROM (SELECT * FROM user WHERE 1782 LIKE 1782) WHERE 999=122 could become SELECT admins FROM (SELECT * FROM user WHERE a1H9 LIKE a1H9) WHERE 999=122

Parameters **payload** –

wafamole.payloadfuzzer.sqlfuzzer.**spaces_to_comments**(*payload*)

wafamole.payloadfuzzer.sqlfuzzer.**spaces_to_whitespaces_alternatives**(*payload*)

wafamole.payloadfuzzer.sqlfuzzer.**swap_int_repr**(*payload*)

wafamole.payloadfuzzer.sqlfuzzer.**swap_keywords**(*payload*)

CHAPTER 2

Running WAF-A-MoLE

2.1 Prerequisites

- numpy
- keras
- scikit-learn
- joblib
- sqlparse
- networkx
- Click

2.2 Setup

```
pip install -r requirements.txt
```

2.3 Sample Usage

You can evaluate the robustness of your own WAF, or try WAF-A-MoLE against some example classifiers. In the first case, have a look at the Model class. Your custom model needs to implement this class in order to be evaluated by WAF-A-MoLE. We already provide wrappers for *sci-kit learn* and *keras* classifiers that can be extend to fit your feature extraction phase (if any).

2.3.1 Help

```
wafamole --help
```

```
Usage: wafamole [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  evade  Launch WAF-A-MoLE against a target classifier.
```

```
wafamole evade --help
```

```
Usage: wafamole evade [OPTIONS] MODEL_PATH PAYLOAD

  Launch WAF-A-MoLE against a target classifier.

Options:
  -T, --model-type TEXT     Type of classifier to load
  -t, --timeout INTEGER     Timeout when evading the model
  -r, --max-rounds INTEGER  Maximum number of fuzzing rounds
  -s, --round-size INTEGER  Fuzzing step size for each round (parallel fuzzing
                            steps)
  --threshold FLOAT         Classification threshold of the target WAF [0.5]
  --random-engine TEXT      Use random transformations instead of evolution
                            engine. Set the number of trials
  --output-path TEXT        Location were to save the results of the random
                            engine. NOT USED WITH REGULAR EVOLUTION ENGINE
  --help                    Show this message and exit.
```

2.3.2 Evading example models

We provide some pre-trained models you can have fun with, located in wafamole/models/custom/example_models. The classifiers we used are listed in the table below.

Classifier name	Algorithm
WafBrain	Recurrent Neural Network
Token-based	Naive Bayes
Token-based	Random Forest
Token-based	Linear SVM
Token-based	Gaussian SVM
SQLiGoT - Directed Proportional	Gaussian SVM
SQLiGoT - Directed Unproportional	Gaussian SVM
SQLiGoT - Undirected Proportional	Gaussian SVM
SQLiGoT - Undirected Unproportional	Gaussian SVM

2.3.2.1 WAF-BRAIN - Recurrent Neural Newtork

Bypass the pre-trained WAF-Brain classifier using a `admin' OR 1=1#` equivalent.

```
wafamole evade --model-type waf-brain wafamole/models/custom/example_models/waf-brain.
↪h5  "admin' OR 1=1#"
```

2.3.2.2 Token-based - Naive Bayes

Bypass the pre-trained token-based Naive Bayes classifier using a `admin' OR 1=1#` equivalent.

```
wafamole evade --model-type token wafamole/models/custom/example_models/nb_trained.
↪dump  "admin' OR 1=1#"
```

2.3.2.3 Token-based - Random Forest

Bypass the pre-trained token-based Random Forest classifier using a `admin' OR 1=1#` equivalent.

```
wafamole evade --model-type token wafamole/models/custom/example_models/rf_trained.
↪dump  "admin' OR 1=1#"
```

2.3.2.4 Token-based - Linear SVM

Bypass the pre-trained token-based Linear SVM classifier using a `admin' OR 1=1#` equivalent.

```
wafamole evade --model-type token wafamole/models/custom/example_models/lin_svm_
↪trained.dump  "admin' OR 1=1#"
```

2.3.2.5 Token-based - Gaussian SVM

Bypass the pre-trained token-based Gaussian SVM classifier using a `admin' OR 1=1#` equivalent.

```
wafamole evade --model-type token wafamole/models/custom/example_models/gauss_svm_
↪trained.dump  "admin' OR 1=1#"
```

2.3.2.6 SQLiGoT

Bypass the pre-trained SQLiGOT classifier using a `admin' OR 1=1#` equivalent. Use **DP**, **UP**, **DU**, or **UU** for (respectivly) Directed Proportional, Undirected Proportional, Directed Unproportional and Undirected Unproportional.

```
wafamole evade --model-type DP wafamole/models/custom/example_models/graph_directed_
↪proportional_sqligot "admin' OR 1=1#"
```

BEFORE LAUNCHING EVALUATION ON SQLiGoT

These classifiers are more robust than the others, as the feature extraction phase produces vectors with a more complex structure, and all pre-trained classifiers have been strongly regularized. It may take hours for some variants to produce a payload that achieves evasion (see Benchmark section).

CHAPTER 3

Benchmark

We evaluated WAF-A-MoLE against all our example models.

The plot below shows the time it took for WAF-A-MoLE to mutate the `admin' OR 1=1#` payload until it was accepted by each classifier as benign.

On the *x* axis we have time (in seconds, logarithmic scale). On the *y* axis we have the *confidence* value, i.e., how sure a classifier is that a given payload is a SQL injection (in percentage).

Notice that being "50% sure" that a payload is a SQL injection is equivalent to flipping a coin. This is the usual classification threshold: if the confidence is lower, the payload is classified as benign.

Fig. 1: Benchmark over time

Experiments were performed on DigitalOcean Standard Droplets.

CHAPTER 4

Contribute

Questions, bug reports and pull requests are welcome.

In particular, if you are interested in expanding this project, we look for the following contributions:

1. New WAF adapters
2. New mutation operators
3. New search algorithms

CHAPTER 5

Team

- Luca Demetrio - CSecLab, DIBRIS, University of Genova
- Andrea Valenza - CSecLab, DIBRIS, University of Genova
- Gabriele Costa - SysMA, IMT Lucca
- Giovanni Lagorio - CSecLab, DIBRIS, University of Genova

Python Module Index

W

Index