WAF-A-MoLE

Evading Web Application Firewalls through Adversarial Machine Learning

Luca Demetrio, Andrea Valenza, Giovanni Lagorio

Università di Genova

{luca.demetrio,andrea.valenza}@dibris.unige.it

giovanni.lagorio@unige.it

Gabriele Costa

IMT School for Advanced Studies

gabriele.costa@imtlucca.it

@zangobot
@AvalZ_

Contributions

- WAF-A-MoLE: a tool to produce adversarial examples against WAFs
- State-of-the-art ML-based WAFs benchmark and bypass
- Dataset of benign and malicious SQL queries

SQL Injections (SQLi)

- Injects arbitrary behaviour
- OWASP Top Ten #1 since 2010
- Should be sanitized

```sql
SELECT * FROM users
WHERE username='admin' OR 1=1#'
and password = 'zenhack'
```

Web Application Firewalls

- Inspects incoming payloads to detect (and block) attacks
- Traditionally signature-based (e.g., white/black lists, regular expressions)
- WAFs treat the symptoms, not the disease

WAFs powered by Machine Learning

- Learns the *model* from the *syntax* of standard attacks (e.g., ZAP, sqlmap)
- Uses *proximity* instead of exact rule matching
- Assigns a *confidence* score to input payloads

- WAF-Brain[1], End-to-end recurrent neural network
- Token-based detectors[2]
- SQLiGoT[3], graph-based algorithm

1 BBVA/waf-brain: WAF-Brain - the clever and efficient Firewall for the Web (https://github.com/BBVA/waf-brain)

2 "Classification of malicious web code by machine learning", *Komiya, Ryohei and Paik, Incheon and Hisada, Masayuki*, iCAST 2011
"SQL Injection detection using machine learning", *Joshi, Anamika and Geetha, V*, ICCICCT 2014

3 "SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM" *Kar, Debabrata and Panigrahi, Suvasini and Sundararajan, Srikanth,* Computers & Security, 2016

Training and Benchmark

		A	R	P
ModSecurity CSR	Paranoia 1/2	86.10%	86.10%	100%
	Paranoia 3/4	91.85%	91.85%	100%
	Paranoia 5	96.46%	96.46%	100%
WAF-Brain	RNN	98.27%	96.73	99.8%
Token-based	Naive Bayes	50.16%	98.71%	50.08%
	Random forest	98.33%	98.33%	100%
	Linear SVM	98.75%	98.76%	100%
	Gaussian SVM	97.82%	97.82%	100%
SQLiGoT	Dir. Prop.	90.61%	97.30%	85.82%
	Undir. Prop.	96.38%	97.31%	95.54%
	Dir. Unprop.	90.52%	97.12%	85.80%
	Undir. Unprop.	96.25%	97.05%	95.53%

Are ML WAFs effective?

Trained on our dataset

Performances compared to signature based WAF (ModSecurity CSR)

Mangling SQLi Syntax

```
admin' OR 1=1#
```

≡

```
admin' OR 0X1=1 or 0x726!=0x726 OR
0x1Dd not IN/*(seleCt 0X0)>c^Bj
>N]*/ ((SeLeCT 476) ,(SELECT(SElEct
477)) ,0X1de) oR 8308 noT lIkE
8308\x0c AnD truE OR 'FZ6/q' LiKE
'fz6/qI' anD TRUE anD
'>U'!='>uz'#t'%'03;Nd
```

Same attack, different syntax

Adversarial Machine Learning

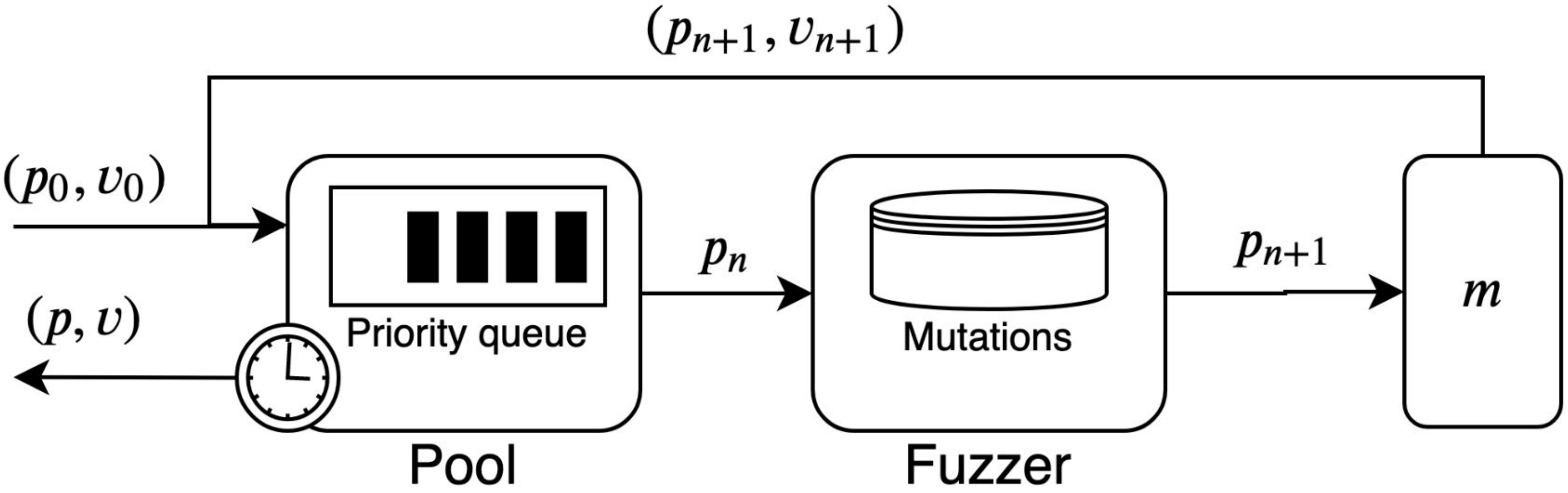
- Introduces the presence of an **adversary**
- Looks for blind spots in **corner-cases**
- **Evasion attack:** malicious payloads seen as legit
- **Constraint:** preserves attack semantics

$$x^* = \underset{x,}{\operatorname{argmin}}\, D(f(x), c_t)$$

subject to C(x)

WAF-A-MoLE

Gray-box: only uses the confidence score assigned by the WAF
Efficient: prioritizes most promising (i.e., fittest) candidates
Plug-and-play: can be used against every SQLi detector (using the proper driver)



Mutational Fuzzing

$$
\begin{aligned}
&\textbf{input}:\ \text{Model } m,\ \text{Payload } p_0,\ \text{Threshold } t\\
&\textbf{output}:\ \textbf{head}(Q)\\
\\
&Q := \text{create\_priority\_queue}()\\
&v := \textbf{classify}(m,\ p_0)\\
&\textbf{enqueue}(Q,\ p_0,\ v)\\
&\textbf{while}\ v > t\\
&\quad p := \textbf{mutate}(\textbf{head}(Q))\\
&\quad v := \textbf{classify}(m,\ p)\\
&\quad \textbf{enqueue}(Q,\ p,\ v)
\end{aligned}
$$

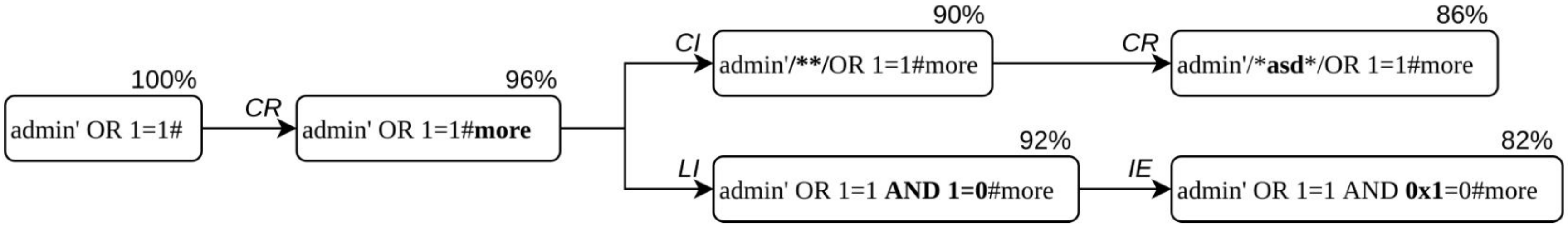
Randomly *mutates* the payload

Selects the best variants

Favors the survival of fittest candidates with a priority queue

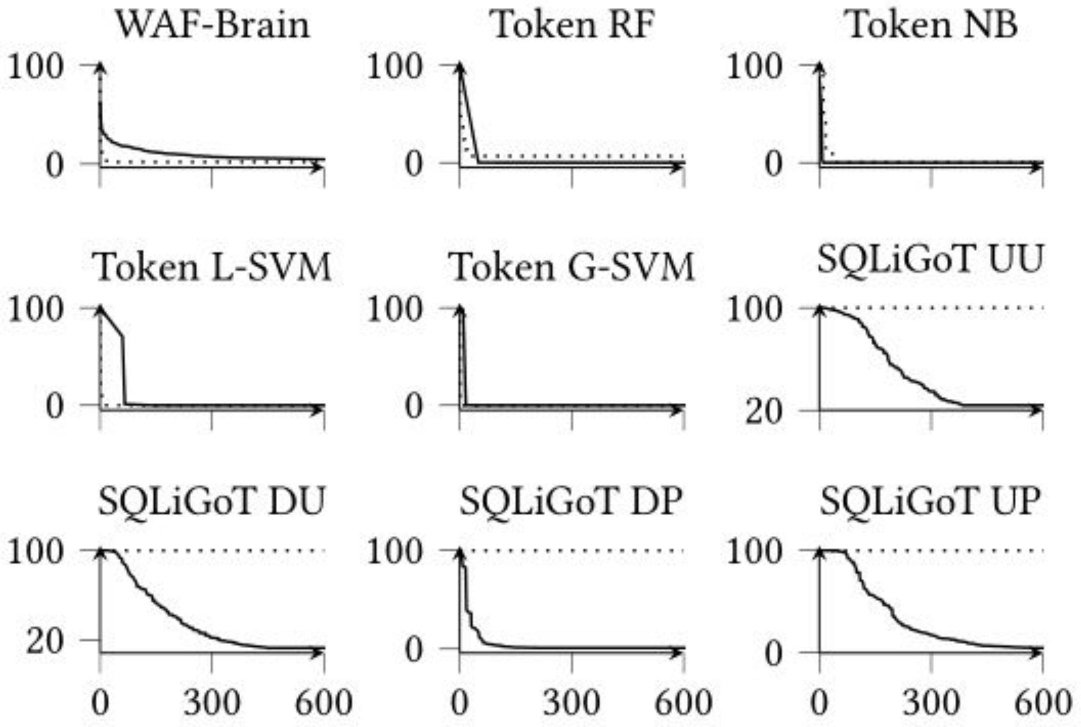
Mutation Operators and Example

Operator	Short definition	Example
Case Swapping	$CS(\ldots a \ldots B \ldots) \to \ldots A \ldots b \ldots$	CS(`admin' OR 1=1#`) → `ADmIn' oR 1=1#`
Whitespace Substitution	$WS(\ldots k_1 k_2 \ldots) \to \ldots k_1 \textvisiblespace k_2 \ldots$	WS(`admin' OR 1=1#`) → `admin'\n OR \t 1=1#`
Comment Injection	$CI(\ldots k_1 k_2 \ldots) \to \ldots k_1 /\ast\ast/ k_2 \ldots$	CI(`admin' OR 1=1#`) → `admin'/**/OR 1=1#`
Comment Rewriting	$CR(\ldots /\ast s_0 \ast/ \ldots \# s_1) \to \ldots /\ast s_0' \ast/ \ldots \# s_1'$	CR(`admin'/**/OR 1=1#`) → `admin'/*abc*/OR 1=1#xyz`
Integer Encoding	$IE(\ldots n \ldots) \to \ldots 0x[n]_{16}$	IE(`admin' OR 1=1#`) → `admin' OR 0x1=1#`
Operator Swapping	$OS(\ldots \oplus \ldots) \to \ldots \boxplus \ldots$ (with $\oplus \equiv \boxplus$)	OS(`admin' OR 1=1#`) → `admin' OR 1 LIKE 1#`
Logical Invariant	$LI(\ldots e \ldots) \to \ldots e$ AND $\top \ldots$	LI(`admin' OR 1=1#`) → `admin' OR 1=1 AND 2<>3#`



Confidence over Mutation Rounds

WAF-A-MoLE (line) vs. random fuzzer (dots)



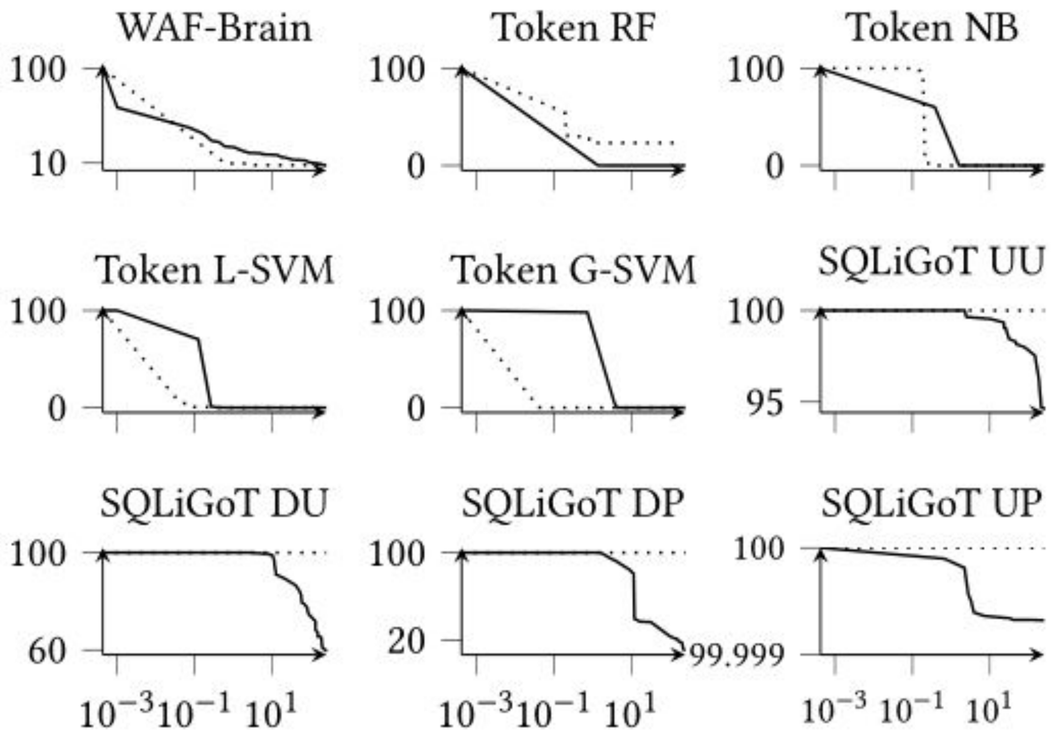
X mutation rounds (1,..,600)
Y WAF confidence (%)

WAF-Brain and Token-based evaded even by **random** (!)

SQLiGoT variants are robust to random mutations, but **vulnerable** to our approach

Confidence over Time

WAF-A-MoLE (line) vs. random fuzzer (dots)



X elapsed time (0,..,100 sec)
Y WAF confidence (%)

WAF-Brain and Token-based evaded in **few seconds**

SQLiGoT variants need more time, especially UP

Conclusions

- The attacker harnesses **infinite** representations of the same payload
- Complexity of features **does not imply** adversarial robustness
- Statistical approaches **may be ineffective** against an aware adversary
- Unexpectedly, **random fuzzing** is sometimes effective
- Guided approach successfully **evades all models**

Thank you for your attention!

- Tool https://github.com/AvalZ/waf-a-mole
- Dataset https://github.com/zangobot/wafamole_dataset

{luca.demetrio,andrea.valenza}@dibris.unige.it

@zangobot
@AvalZ_