ZeroWall: Detecting Zero-Day Web Attacks through Encoder-Decoder Recurrent Neural Networks

Ruming Tang*, Zheng Yang*, Zeyan Li*, Weibin Meng*, Haixin Wang[+],
Qi Li*, Yongqian Sun[#], Dan Pei*, Tao Wei[^], Yanfei Xu[^] and Yan Liu[^]

*Tsinghua University, China

[+] University of Science and Technology Beijing, China

[#]Nankai University, China

[^]Baidu

IEEE International Conference on Computer Communications, 27-30 April 2020 // Beijing, China

WAFs Do Not Capture Zero-Days

- **WAF**s (**W**eb **A**pplication **F**irewalls) are **wildly deployed** in industry, however, such **signature-based** methods are not suitable to detect zero-day attacks.
- Zero-day attacks in general are hard to detect and zero-day Web attacks are particularly challenging because:
  1. have **not been previously seen**
     → most **supervised** approaches are inappropriate
  2. can be carried out by a **single** malicious HTTP request
     → **contextual** information is not helpful
  3. very **rare** within a large number of Web requests
     → **collective** and **statistical** information are not effective

ZeroWall

An **unsupervised** approach, which can **work with an existing WAF in pipeline**, to effectively detecting a zero-day Web attack hidden in **an individual Web request**.

What We Want

- WAF detects those **known** attacks effectively.
  - filter out **known** attacks
- **ZeroWall** detects **unknown** attacks **ignored by WAF rules**.
  - report **new attack patterns** to operators and security engineers to **update WAF rules**.

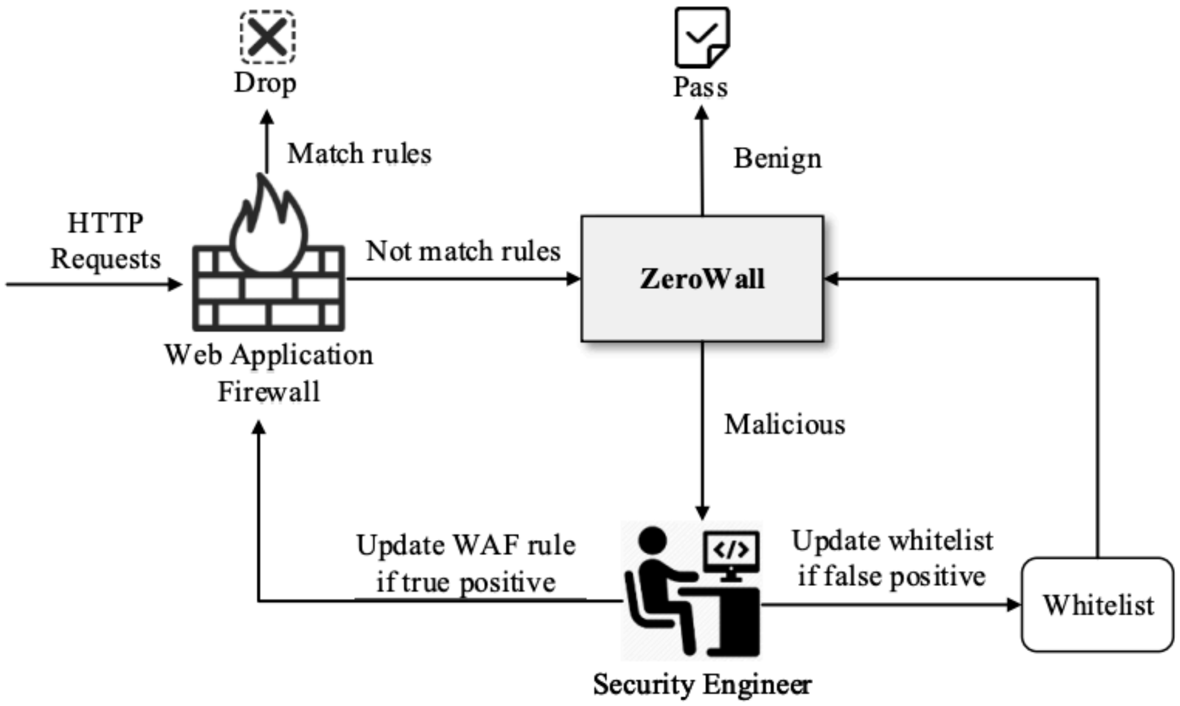
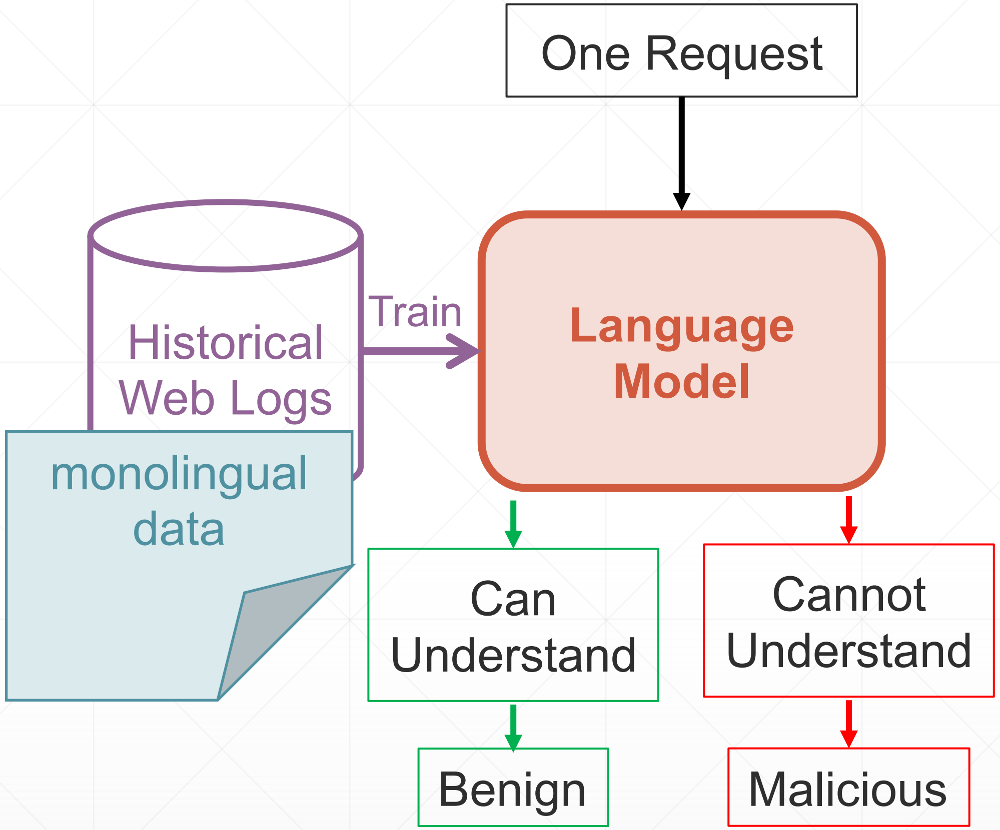


Figure 1: The workflow of *ZeroWall*.

Idea

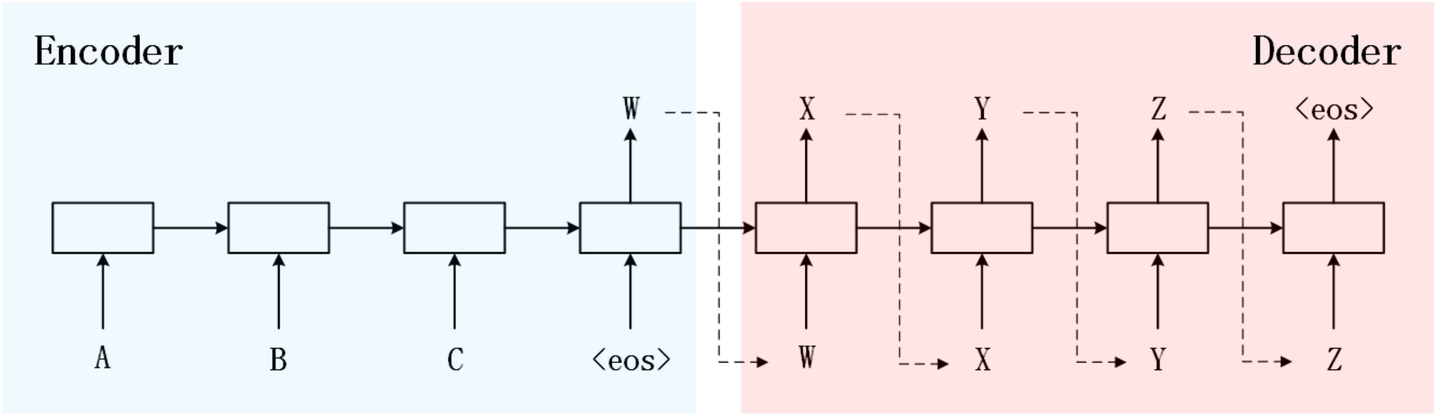
- HTTP request is a **string following HTTP**, and we can consider an HTTP request as one **sentence** in the ***HTTP request language***.
- **Most** requests are **benign**, and **malicious** requests are **rare**.
- Thus, we train a kind of **language model** based on historical logs, to **learn this language** from **benign requests**.



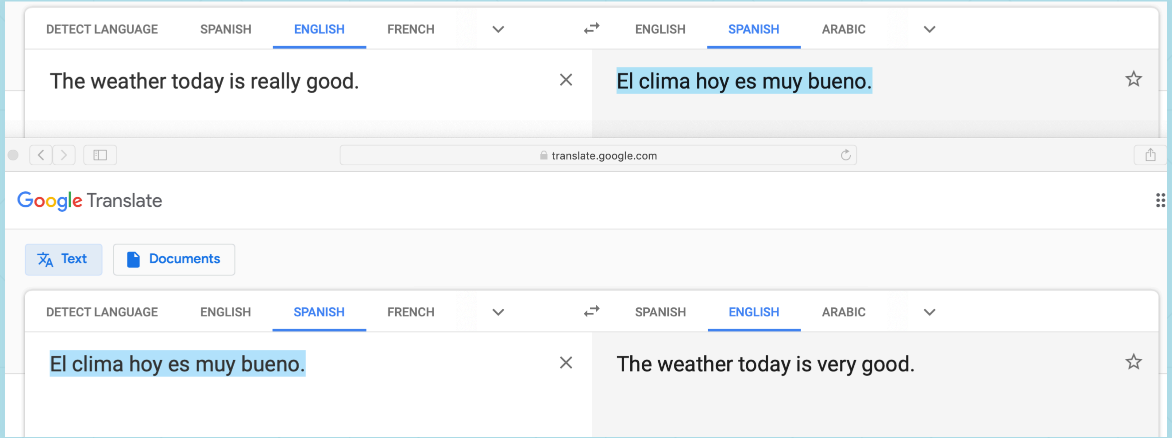
Self-Translate Machine

- How to learn this "**Hyper-TEXT**" language?

- Use **Neural Machine Translation** model to train a **Self-Translate Machine**
  - **Encode** the original request into one *representation*
  - Then **Decode** it back

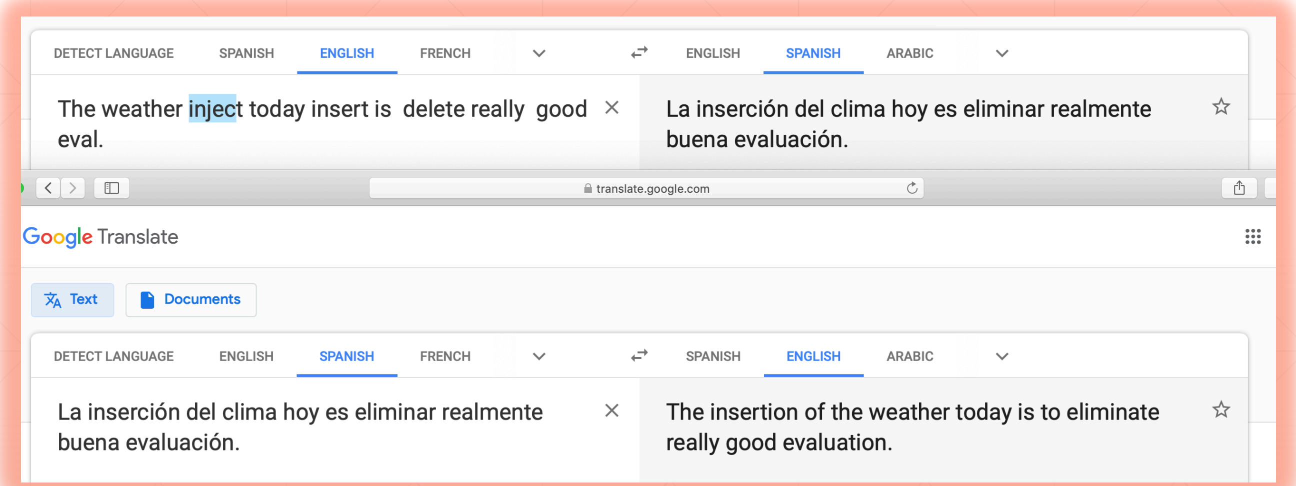


Self-Translate Machine



Self-translation works **well** for **normal** sentences

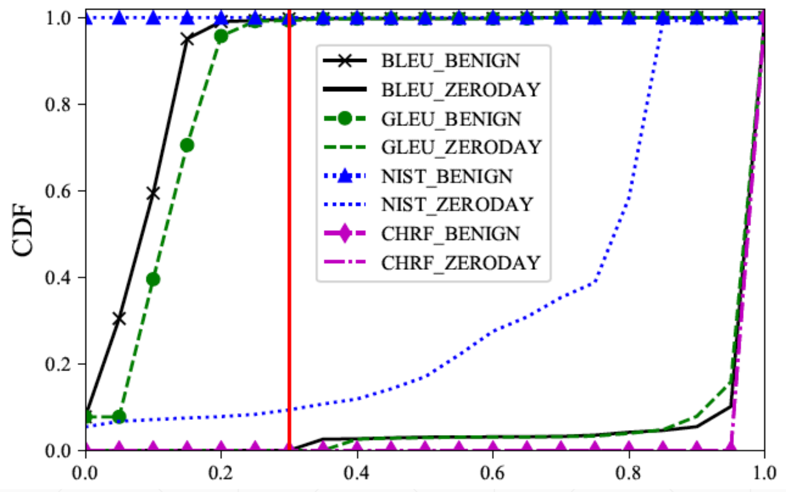
Output **deviates** significantly from the input, when the input is a sentence **not previously seen** in the training dataset of the self-translation models.

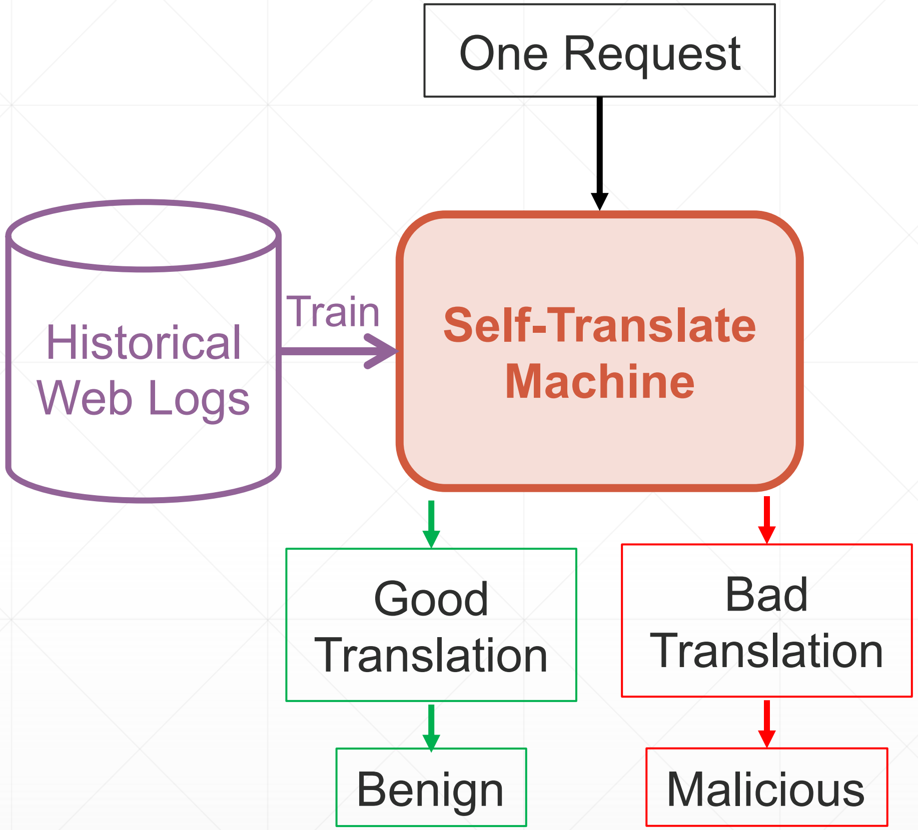


Self-Translate Machine

- **Translation Quality → Anomaly Score**
- How to quantify the self-translation quality (anomaly score)?

→ Use **machine translation metrics**





An attack detection problem → A machine translation quality assessment problem

Self-Translated Sequence

- **Translation Quality → Anomaly Score**

  → Use **BLEU** as an example

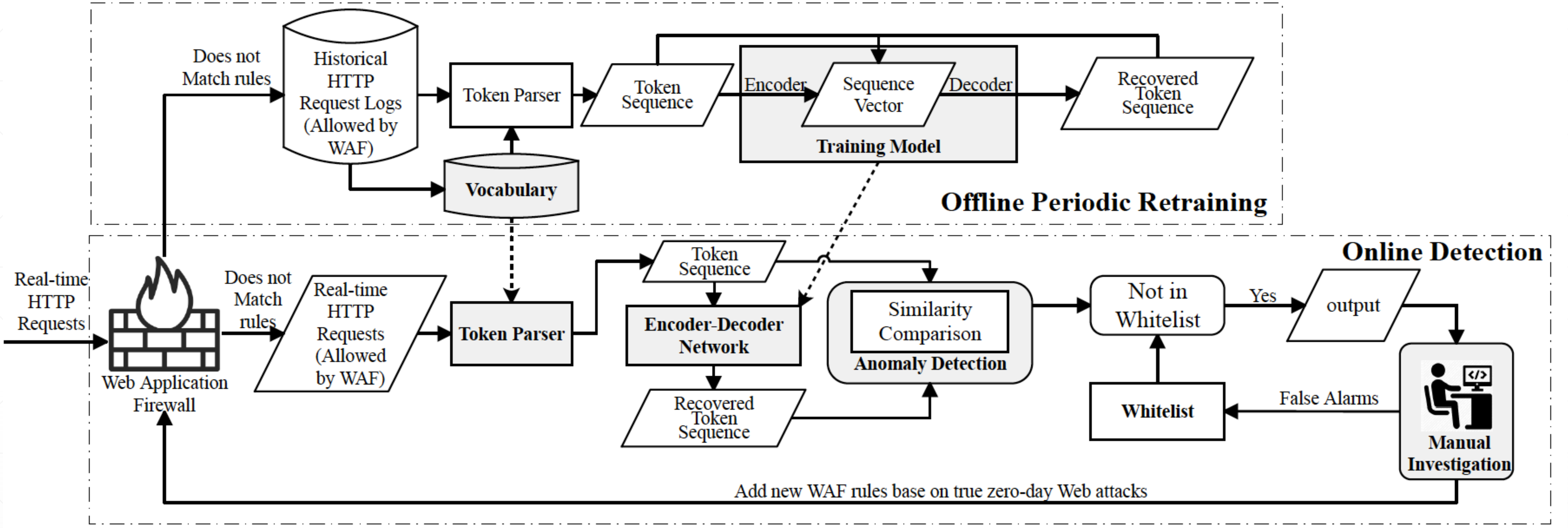
  → **Malicious Score** = $1 - BLEU\_Score$

Original Request	POST http://localhost:8080/tienda1/publico/autenticar.jsp modo=entrar&login=caria&pwd=egipciaca&remember=off&B1=Entrar		
Tokenized	tienda1 publico autenticar jsp modo entrar login _OTHER_ pwd _OTHER_ remember off b1 entrar		
Translated	tienda1 publico autenticar jsp modo entrar login _OTHER_ pwd _OTHER_ remember on b1 entrar		
BLEU	0.8091	**Malicious Score**	0.1909

Original Request	POST http://m.thepaper.cn/admin_UploadDataHandler.ashx ------WebKitFormBoundaryRvkd1dbq3x1OJhUH\x0D\x0AContent-Disposition: form-data; name=\x22uploadify\x22; filename=\x2220170215180046.jpg\x22\x0D\x0A *Content-Type: image/jpeg*\x0D\x0A\x0D\x0A **<%eval request(\x22T\x22)%>**\x0D\x0A------WebKitFormBoundaryRvkd1dbq3x1OJhUH\x0D\x0AContent-Disposition: form-data; name=\x22saveFile\x22\x0D\x0A\x0D\x0At.asp\x0D\x0A------WebKitFormBoundaryRvkd1dbq3x1OJhUH\x0D\x0AContent-Disposition: form-data; name=\x22Upload\x22\x0D\x0A\x0D\x0ASubmit Query\x0D\x0A------WebKitFormBoundaryRvkd1dbq3x1OJhUH--		
Tokenized	_OTHER_ ashx _OTHER_ content disposition form data name uploadify filename _pnum_0_ jpg content type image jpeg eval request onechr _OTHER_ content disposition form data name _OTHER_ onechr asp _OTHER_ content disposition form data name upload submit query _OTHER_		
Translated	_OTHER_ _OTHER_ do php _OTHER_ eval get_magic_quotes_gpc stripslashes _post chr _pnum_0_ chr _pnum_1_ _post chr _pnum_2_ chr _pnum_3+_ z0 _pnum_3+_ ini_set display_errors _pnum_3+_ set_time_limit _pnum_3+_ set_magic_quotes_runtime _pnum_3+_ echo onechr dirname _server script_filename if onechr onechr dirname _server path_translated		
BLEU	0	**Malicious Score**	1.0

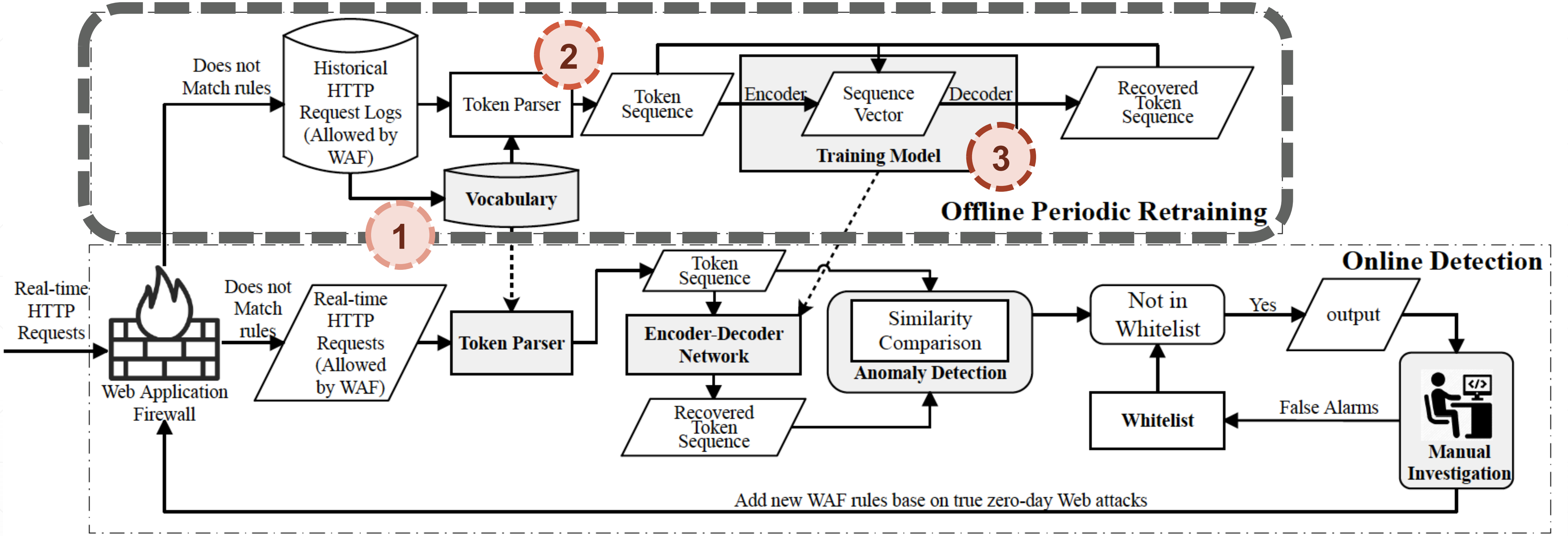
An attack detection problem → A machine translation quality assessment problem

ZeroWall Workflow



- Offline Periodic Retraining
  - Build and update **vocabulary** and re-train the **model**
- Online Detection
  - Detect **anomalies** in real-time requests for **manual investigation**

Offline Training

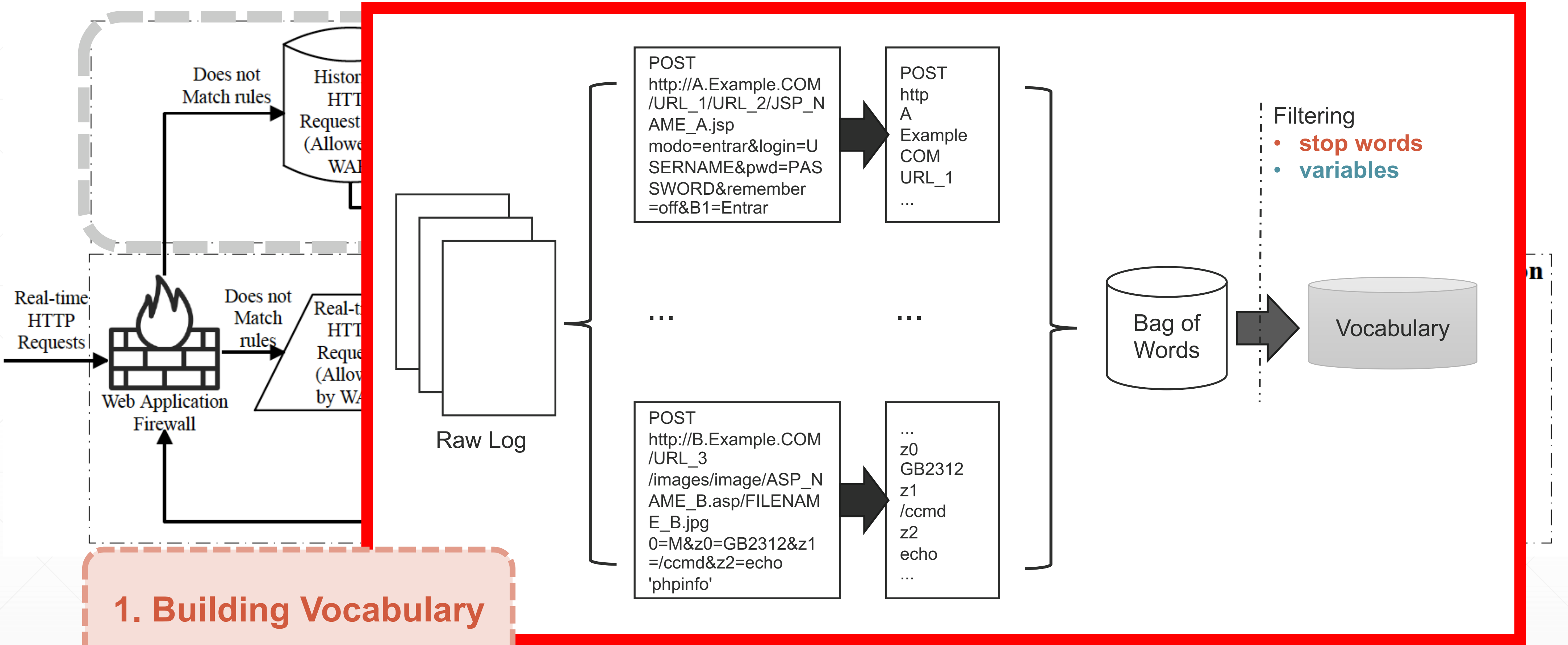


1. Building Vocabulary

2. Parsing

3. Training Model

Offline Training

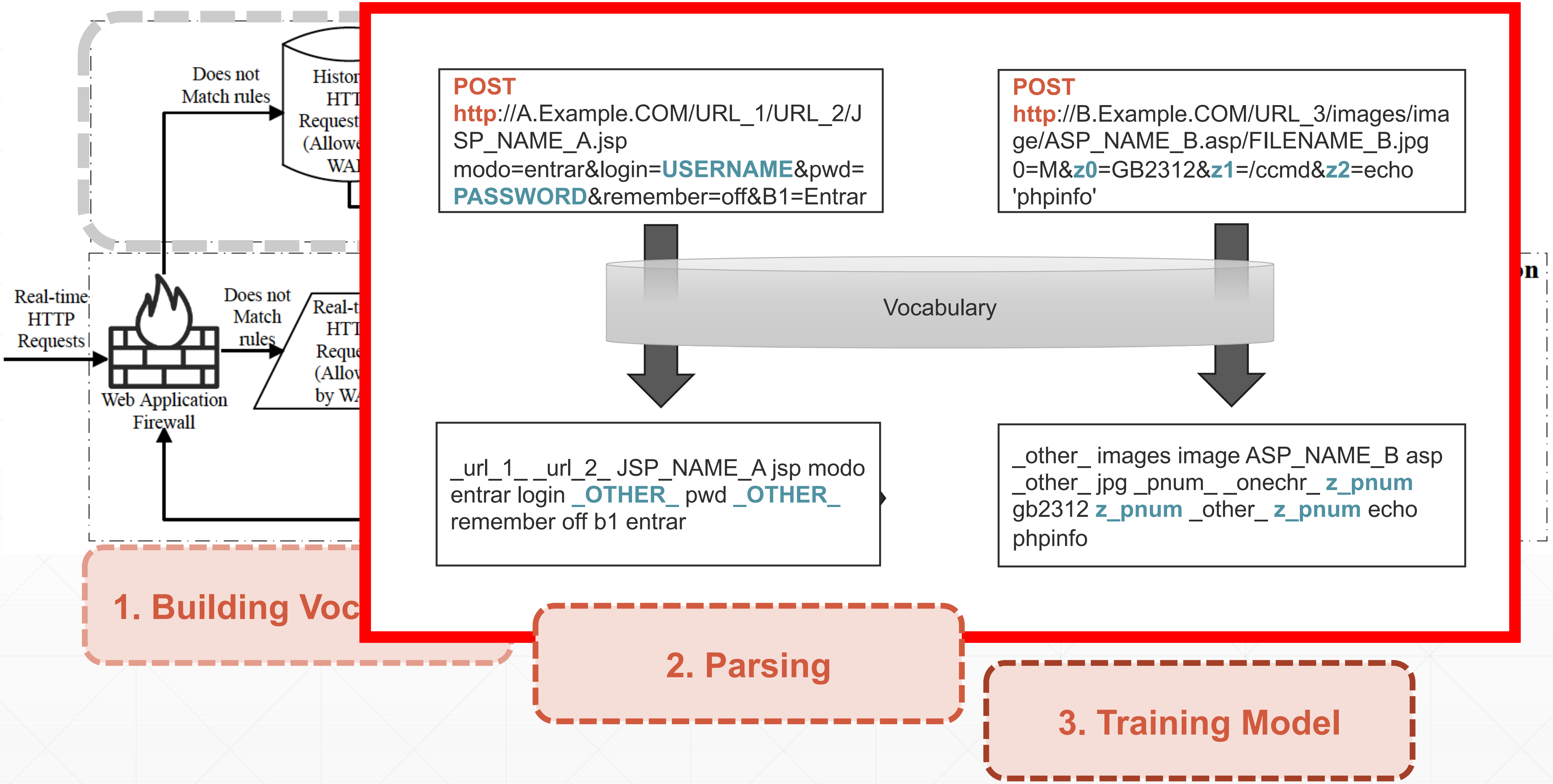


1. Building Vocabulary

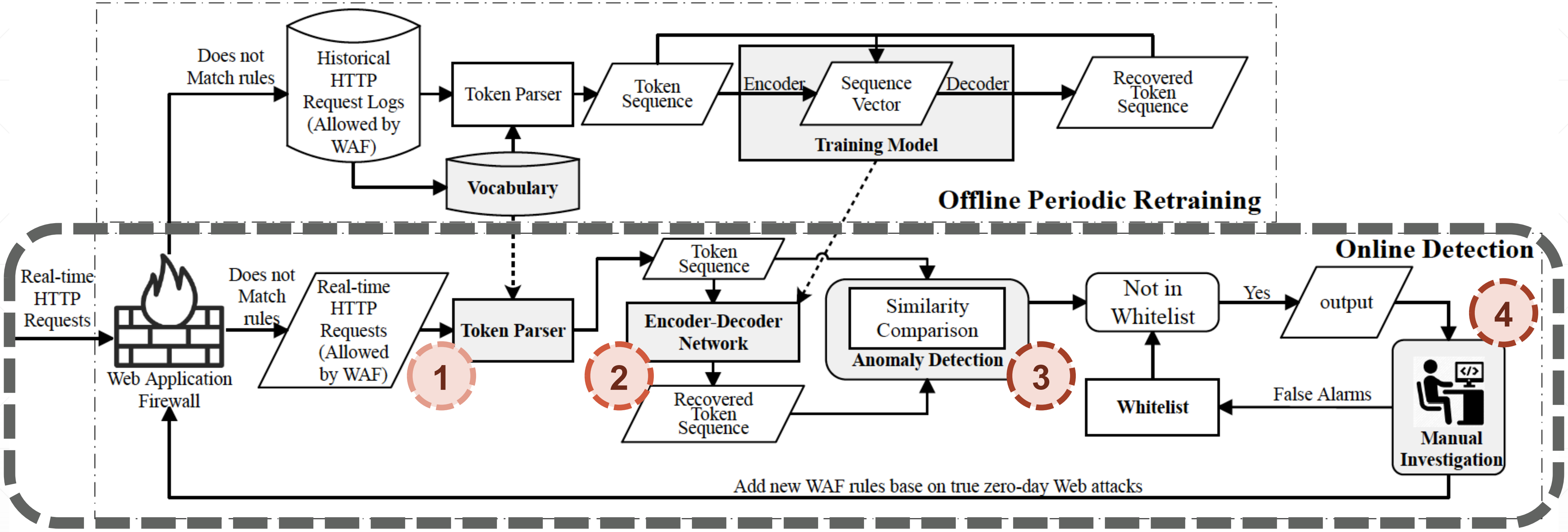
2. Parsing

3. Training Model

Offline Training



Online Detection



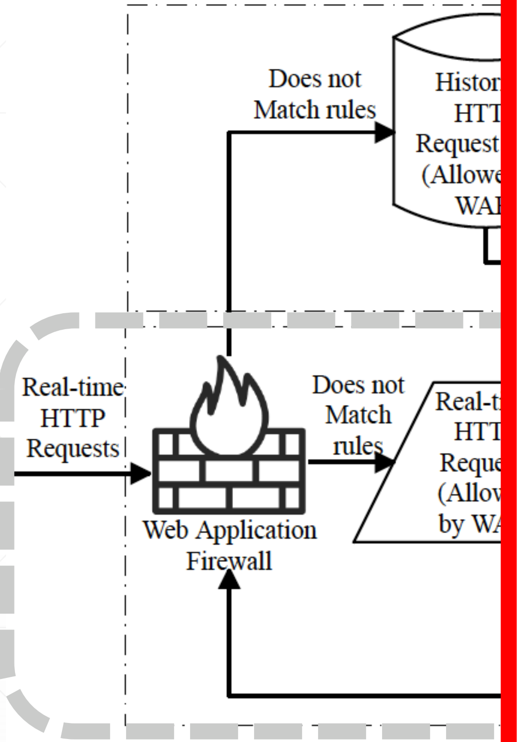
1. Parsing

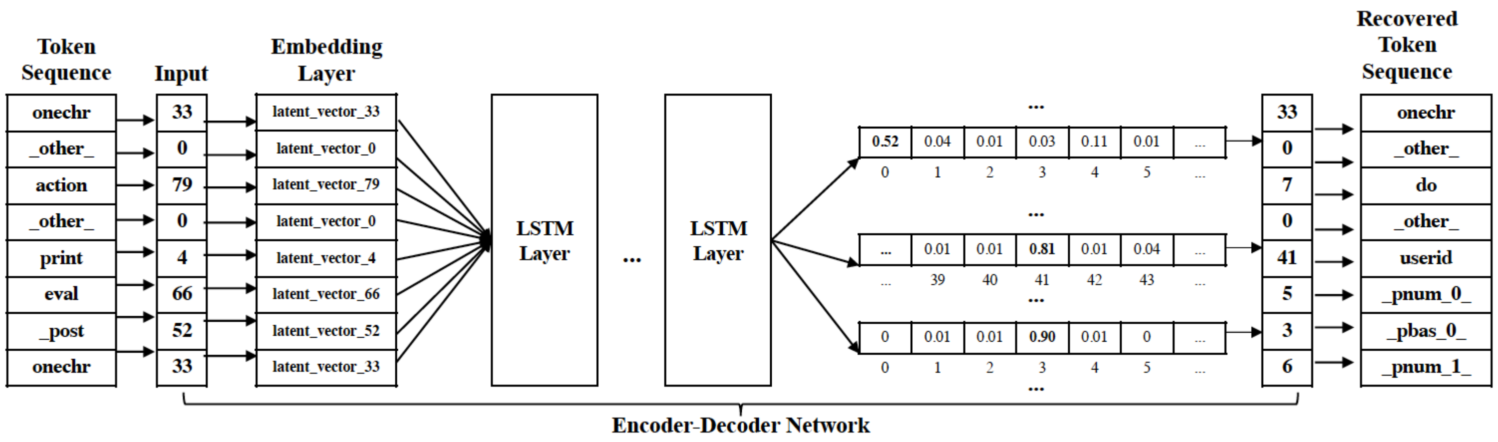
2. Translation

3. Detection

4. Investigation

Online Detection





→ onechr _OTHER_ action _OTHER_ print eval _post onechr

→ onechr _OTHER_ do _OTHER_ userid _pnum_0_ _pbas_0_ _pnum_1_

1. Token Parser

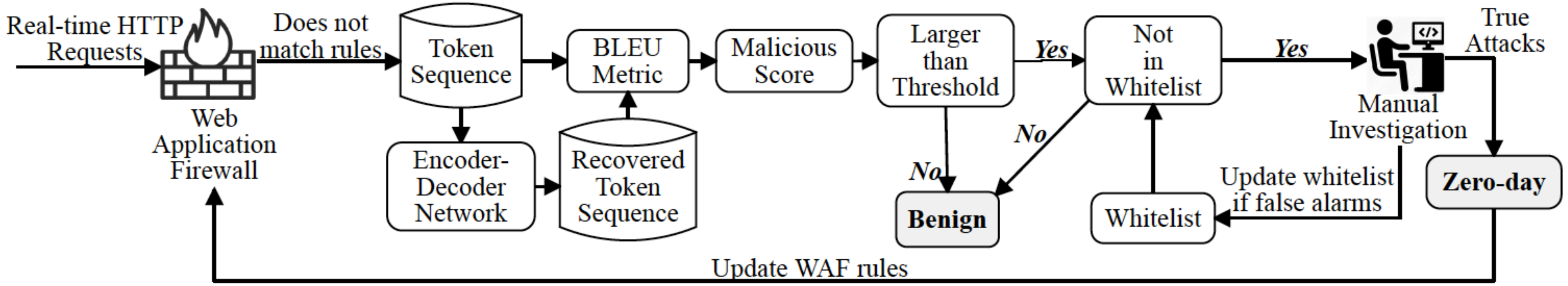
2. Translation

3. Detection

4. Investigation

Online Detection

Compare the original sequence (token sequence) and the translated sequence (recovered token sequence).



1. BLEU Metric
2. Threshold [Larger? Yes → Go to step 3; No → Benign]
3. Check whitelist [Not in whitelist? Yes → Go to step 4; No → Benign]
4. Investigation [True Attacks → Update WAF/IDS ; False Alarms → Update whitelist rules]

1. Token Parser | 2. Translation | 3. Detection | 4. Investigation

Real-World Deployment

- Data Trace:
  - 8 real world trace from an Internet company.
  - Over 1.4 billion requests in a week.
- Overview
  - Captured 28 different types of zero-day attacks, which contribute to 10K of zero-day attack requests in total.
  - False positives: 0~6 per day

#	D-1	D-2	D-3	D-4	D-5	D-6	D-7	D-8	Total
Malicious*	51839	186066	19515	53394	33724	2136811	42088623	90982519	135552491
Zero-Day	25	1118	283	4209	1188	2003	49011	83746	141583
Benign	1576235	3142793	13572827	15618518	31718124	177993528	528158912	534048878	1305829815
Total	1628099	3329977	13592625	15676121	31753036	180132342	570296546	625115143	1441523889
B2M(1)	30.4	16.9	695.5	292.5	940.5	83.3	12.5	5.9	9.6
B2Z(2)	63049.4	2811.1	47960.5	3710.7	26698.8	88863.5	10776.3	6377.0	9223.1

* Known malicious filtered by WAF. (1) Ratio of **B**enign to **M**alicious (in WAF); (2) Ratio of **B**enign to **Z**ero-Day

Baselines & Labels

- Unsupervised Approaches
  - SAE (stacked auto-encoder), HMM and DFA (Deterministic Finite Automata)
  - Use data filtered by WAF as training set.
- Supervised Approaches
  - CNN, RNN and DT (decision tree)
  - Use all data (allowed/dropped) as training set and WAF results as labels.

Evaluation Results

Trace	Approach	Precision	Recall	F1-Score
D-1	***ZeroWall***	**0.9855**	**1.0000**	**0.9889**
#WAF-Malicious: 51,839	*DT-Token*	0.0010	1.0000	0.0019
#Zero-Day Attacks: 25	*CNN-Token*	0.0010	1.0000	0.0019
#Benign: 1,576,235	*RNN-Token*	0.0000	1.0000	0.0000
#Total: 1,628,099	*SAE*	0.0001	1.0000	0.0002
B2M: 30.4	*Hmmpayl*	0.0000	0.0000	0.0000
B2Z: 63049.4	*DFA*	0.0000	1.0000	0.0000
D-2	***ZeroWall***	**1.0000**	**1.0000**	**1.0000**
#WAF-Malicious:186,066	*DT-Token*	0.0547	0.3712	0.0931
#Zero-Day: 1,118	*CNN-Token*	0.3300	0.7784	0.4593
#Benign: 3,142,793	*RNN-Token*	0.0005	0.9760	0.0010
#Total: 3,329,977	*SAE*	0.0005	0.9820	0.0010
B2M: 16.9	*Hmmpayl*	0.0000	0.0000	0.0000
B2Z: 2811.1	*DFA*	0.0004	1.0000	0.0008
D-3	***ZeroWall***	**0.9925**	**0.9687**	**0.9805**
#WAF-Malicious: 19,515	*DT-Token*	0.7388	0.2463	0.3658
#Zero-Day: 283	*CNN-Token*	0.4230	0.6376	0.5039
#Benign: 13,572,827	*RNN-Token*	0.0000	0.9999	0.0001
#Total: 13,592,625	*SAE*	0.0015	0.9130	0.0030
B2M: 695.5	*Hmmpayl*	0.0000	0.0000	0.0000
B2Z: 47960.5	*DFA*	0.0000	1.0000	0.0001
D-4	***ZeroWall***	**0.9879**	**1.0000**	**0.9939**
#WAF-Malicious: 53,394	*DT-Token*	0.0001	1.0000	0.0002
#Zero-Day: 4,209	*CNN-Token*	0.0001	1.0000	0.0002
#Benign: 15,618,518	*RNN-Token*	0.0008	1.0000	0.0016
#Total: 15,676,121	*SAE*	1.0000	0.0000	0.0000
B2M: 292.5	*Hmmpayl*	0.0000	0.0000	0.0000
B2Z: 3710.7	*DFA*	0.0001	1.0000	0.0002
D-5	***ZeroWall***	**0.9928**	**1.0000**	**0.9964**
#WAF-Malicious: 33,724	*DT-Token*	0.2497	0.0082	0.0153
#Zero-Day: 1,188	*CNN-Token*	0.6567	0.5410	0.5883
#Benign: 31,718,124	*RNN-Token*	0.9988	0.0328	0.0629
#Total: 31,753,036	*SAE*	0.0000	0.0492	0.0000
B2M: 940.5	*Hmmpayl*	—	—	—
B2Z: 26698.8	*DFA*	0.0001	1.0000	0.0001
D-6	***ZeroWall***	**1.0000**	**0.9897**	**0.9948**
#WAF-Malicious:2,136K	*DT-Token*	0.1733	0.0365	0.0576
#Zero-Day: 2,003	*CNN-Token*	0.0204	0.0590	0.0269
#Benign: 177,993,528	*RNN-Token*	0.0000	1.0000	0.0000
#Total: 180,132,342	*SAE*	0.0001	0.1461	0.0001
B2M: 83.3	*Hmmpayl*	-	-	-
B2Z: 88863.5	*DFA*	0.0000	1.0000	0.0000
D-7	***ZeroWall***	**0.9943**	**1.0000**	**0.9971**
#WAF-Malicious:42,088K	*DT-Token*	0.0874	0.0267	0.0377
#Zero-Day: 49,011	*CNN-Token*	0.8094	0.3027	0.4366
#Benign: 528,158,912	*RNN-Token*	0.6857	0.5608	0.6120
#Total: 570,296,546	*SAE*	0.0001	0.5691	0.0002
B2M: 12.5	*Hmmpayl*	-	-	-
B2Z: 10776.3	*DFA*	0.0001	1.0000	0.0002
D-8	***ZeroWall***	**0.9966**	**0.9983**	**0.9974**
#WAF-Malicious:90,982K	*DT-Token*	0.2036	0.3054	0.2396
#Zero-Day: 83,746	*CNN-Token*	0.2525	0.0275	0.0479
#Benign: 534,048,878	*RNN-Token*	0.5237	0.0718	0.1242
#Total: 625,115,143	*SAE*	0.0008	0.3476	0.0017
B2M: 5.9	*Hmmpayl*	-	-	-
B2Z: 6377.0	*DFA*	0.0000	1.0000	0.0005

A Zero-Day Case

- These attack is detected by **ZeroWall**, **CNN** and **RNN**.
- **WAF** are usually based on **keywords**, e.g., eval, request, select and execute.
- **ZeroWall** is based on the "**understanding**" of benign requests. The structure of this zero-day attack request is more like a programming language.

...
searchword=d&**order=}{end if}{if:1)print_r(**
$_POST[func]($_POST[cmd]));//}
{end if}&func=assert&cmd=phpinfo();

Token Sequence: search php searchtype _pnum_0_ _OTHER_ onechr order end if if _pnum_1_ _OTHER_ _post _OTHER_ _post cmd end if _OTHER_ assert cmd phpinfo

contains **none** of **WAF keywords**

overlap with tokens in training set for **CNN** and **RNN**

1	plus ad_js php aid _pnum_0_ onechr assert _pnum_1_ execute execute function bd byval onechr for onechr _pnum_2_ to len onechr step _pnum_3+_ onechr mid onechr _pnum_3+_ if isnumeric mid onechr _pnum_3+_ then execute bd bd chr onechr else execute bd bd chr onechr mid onechr _pnum_3+_ onechr _pnum_3+_ **end if** chr _pnum_3+_ next end function response write execute on error resume next bd _phex_0_ response write response end
2	preview php _OTHER_ php **assert** _OTHER_ onechr
3	lib _OTHER_ module inc php _OTHER_ eval _OTHER_ onechr class _OTHER_ onechr **phpinfo**
4	cms _OTHER_ uploads _OTHER_ php id **assert** _OTHER_ eval base64_decode **_post** z0 z0 _pbas_0_
5	myship php cmd eval base64_decode **_post** z0 z0 _pbas_0_

Whitelist

- To mitigate **False Alarms**, we add **whitelist** to our approach.
- The **numbers of whitelist rules** refer to how many whitelist rules are added each day, based on the FPs labeled on that day. (No rules applied on 0602 since it is the first day of testing set.)
- The results shows that the whitelist **reduces the number of FPs with low overhead** (numbers of rules are very small).
- Based on these results, we believe ZeroWall is practical in real-world deployment.

Date	Precision		# of FP		# of white-list rules
	No WL	WL	No WL	WL	
0602	0.9972	-	16	-	5
0603	0.9643	0.9753	222	152	3
0604	0.9580	0.9999	310	1	1
0605	0.9731	0.9944	320	65	6
0606	0.9845	0.9993	121	5	1
0607	0.9672	1.0000	194	0	0

Overhead

- Training and testing speed with and without hash table (requests/s)

Trace	Incoming Requests	Training		Testing	
		No Hash	Hash	No Hash	Hash
D-1	2.60	1.09	256.89	229.24	39634.40
D-2	5.19	3.72	202.13	785.75	65556.80
D-3	22.44	7.09	835.43	514.33	50420.17
D-4	25.83	5.42	1014.67	305.42	50913.24
D-5	52.45	12.48	1046.55	414.86	38132.88
D-6	294.30	1.47	4001.95	70.04	176255.90
D-7	873.36	3.23	4262.48	53.77	88989.06
D-8	883.16	6.67	6389.23	142.29	106692.90

*The incoming requests refer to the average number of requests received by the customer per second.

Intel(R) Xeon(R) Gold 6148 CPU 2.40GHz * 2
512GB RAM

Summary

- Present a zero-day web attack detection system **ZeroWall**
  - **Augmenting** existing **signature-based WAFs**
  - Use **Encoder-Decoder Network** to learn patterns from normal requests
  - Use **Self-Translate Machine** & **BLEU Metric**

An attack detection problem → A machine translation quality assessment problem

- **Deployed** in the wild
  - Over **1.4** billion requests
  - Captured **28** different types of zero-day attacks (**10K** of zero-day attack requests)
  - Low overhead

Thanks!
And Questions

Ruming Tang: trm14@mails.tsinghua.edu.cn