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 (Web Application Firewalls) 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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokenized</td>
<td>tienda1 publico autenticar jsp modo entrar login <em>OTHER</em> pwd <em>OTHER</em> remember off b1 entrar</td>
</tr>
<tr>
<td>Translated</td>
<td>tienda1 publico autenticar jsp modo entrar login <em>OTHER</em> pwd <em>OTHER</em> remember on b1 entrar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BLEU</th>
<th>Malicious Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8091</td>
<td>0.1909</td>
</tr>
</tbody>
</table>

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

FILTERING
• stop words
• variables

RAW LOG

Bag of Words

Vocabulary
Offline Training

1. Building Vocabulary

2. Parsing

3. Training Model
Online Detection

1. Parsing
2. Translation
3. Detection
4. Investigation
Online Detection

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

→ onechr _OTHER_ action _OTHER_ print eval _post onechr

→ onechr _OTHER_ do _OTHER_ userid _pnum_0 _pbas_0 _pnum_1_
Online Detection

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

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

- Real-time HTTP Requests
- Web Application Firewall
- Does not match rules
- Token Sequence
- Encoder-Decoder Network
- Recovered Token Sequence
- BLEU Metric
- Malicious Score
- Larger than Threshold
- Not in Whitelist
- Benign
- Manual Investigation
- True Attacks

1. **BLEU Metric**
   - [ Larger? Yes \(\rightarrow\) Go to step 3; No \(\rightarrow\) Benign ]

2. **Threshold**
   - [ Larger? Yes \(\rightarrow\) Go to step 3; No \(\rightarrow\) Benign ]

3. **Check whitelist**
   - [ Not in whitelist? Yes \(\rightarrow\) Go to step 4; No \(\rightarrow\) Benign ]

4. **Investigation**
   - [ True Attacks \(\rightarrow\) Update WAF/IDS; False Alarms \(\rightarrow\) Update whitelist rules ]

Update WAF rules
### 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

### #

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

* Known malicious filtered by WAF.  (1) Ratio of Benign to Malicious (in WAF); (2) Ratio of Benign to Zero-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

<table>
<thead>
<tr>
<th>Trace</th>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D-1</strong></td>
<td>ZeroWall</td>
<td>0.9855</td>
<td>1.0000</td>
<td>0.9889</td>
</tr>
<tr>
<td>#WAF-Malicious: 51,839</td>
<td>DT-Token</td>
<td>0.0010</td>
<td>1.0000</td>
<td>0.0019</td>
</tr>
<tr>
<td>#Zero-Day Attacks: 25</td>
<td>CNN-Token</td>
<td>0.0010</td>
<td>1.0000</td>
<td>0.0019</td>
</tr>
<tr>
<td>#Benign: 1,576,235</td>
<td>RNN-Token</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>#Total: 1,628,099</td>
<td>SAE</td>
<td>0.0001</td>
<td>1.0000</td>
<td>0.0002</td>
</tr>
<tr>
<td>B2M: 30.4</td>
<td></td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>B2Z: 63049.4</td>
<td></td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

| **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 | | 0.0000   | 0.0000 | 0.0000  |
| B2Z: 2811.1 | | 0.0004   | 1.0000 | 0.0000  |

| **D-3** | ZeroWall | 0.9925   | 0.9897 | 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 | | 0.0000   | 0.0000 | 0.0000  |
| B2Z: 47960.5 | | 0.0000   | 1.0000 | 0.0000  |

| **D-4** | ZeroWall | 0.9879 | 1.0000 | 0.9939  |
| #WAF-Malicious: 53,394 | DT-Token | 0.0000 | 1.0000 | 0.0002  |
| #Zero-Day: 4,209 | CNN-Token | 0.0000 | 1.0000 | 0.0002  |
| #Benign: 15,618,518 | RNN-Token | 0.0000 | 1.0000 | 0.0002  |
| #Total: 15,676,212 | SAE | 1.0000 | 0.0000 | 0.0000  |
| B2M: 292.5 | | 0.0000 | 0.0000 | 0.0000  |
| B2Z: 3710.7 | | 0.0001 | 1.0000 | 0.0002  |
A Zero-Day Case

- These attacks are 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_f( 
$POST[func]($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

- overlaps with tokens in training set for CNN and RNN
- contains none of WAF keywords
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.

<table>
<thead>
<tr>
<th>Date</th>
<th>Precision</th>
<th># of FP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No WL</td>
<td>WL</td>
</tr>
<tr>
<td>0602</td>
<td>0.9972</td>
<td>-</td>
</tr>
<tr>
<td>0603</td>
<td>0.9643</td>
<td>0.9753</td>
</tr>
<tr>
<td>0604</td>
<td>0.9580</td>
<td>0.9999</td>
</tr>
<tr>
<td>0605</td>
<td>0.9731</td>
<td>0.9944</td>
</tr>
<tr>
<td>0606</td>
<td>0.9845</td>
<td>0.9993</td>
</tr>
<tr>
<td>0607</td>
<td>0.9672</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
## Overhead

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

<table>
<thead>
<tr>
<th>Trace</th>
<th>Incoming Requests</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Hash</td>
<td>Hash</td>
</tr>
<tr>
<td>D-1</td>
<td>2.60</td>
<td>1.09</td>
<td>256.89</td>
</tr>
<tr>
<td>D-2</td>
<td>5.19</td>
<td>3.72</td>
<td>202.13</td>
</tr>
<tr>
<td>D-3</td>
<td>22.44</td>
<td>7.09</td>
<td>835.43</td>
</tr>
<tr>
<td>D-4</td>
<td>25.83</td>
<td>5.42</td>
<td>1014.67</td>
</tr>
<tr>
<td>D-5</td>
<td>52.45</td>
<td>12.48</td>
<td>1046.55</td>
</tr>
<tr>
<td>D-6</td>
<td>294.30</td>
<td>1.47</td>
<td>4001.95</td>
</tr>
<tr>
<td>D-7</td>
<td>873.36</td>
<td>3.23</td>
<td>4262.48</td>
</tr>
<tr>
<td>D-8</td>
<td>883.16</td>
<td>6.67</td>
<td>6389.23</td>
</tr>
</tbody>
</table>

*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**

- **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

An attack detection problem → A machine translation quality assessment problem

**Thanks! And Questions**

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