Detection of Anomalous HTTP Requests Based on N-gram Models, GHSOMs and DBSCAN

Mikhail Zolotukhin, Timo Hämäläinen

Department of Mathematical Information Technology,
University of Jyväskylä, Jyväskylä

17/09/2013
Motivation

- Nowadays the use of computer technologies, both for work and personal use, is growing with time.
- Unfortunately, computer networks and systems are often vulnerable to different forms of intrusions.
- Web-servers and web-based applications is one of the most popular attack targets.
- It is possible to manipulate HTTP queries and create requests which can corrupt the server or collect confidential information:

  **Example**

  http://example.org/script.php?id=-1+OR+1=1.

One means to ensure the security of web-servers and web-based applications is use of Intrusion Detection Systems (IDS).
Misuse and anomaly detection

Misuse detection:
- The IDS scans the computer system for predefined attack signatures.
- This approach is accurate which makes it successful in commercial intrusion detection.
- This approach can not detect attacks for which it has not been programmed, and, therefore, can not detect zero-day attacks.

Anomaly detection:
- The IDS detects patterns which deviate from established norms, i.e. anomalies.
- The IDS is modelled according to normal behaviour and, therefore, is able to detect zero-day attacks.
- The number of false alerts will probably be increased because not all anomalies are intrusions.
Develop the intrusion detection algorithm which employs the anomaly detection approach.

Train the model of normal behavior even in the case the set of HTTP requests free of attacks is not available.

Detect intrusions with high accuracy rate.

Detect HTTP intrusions in continuously updated web-applications.
HTTP requests

Example


- HTTP queries include name of the resource and several attributes.
- Assume that most requests, which are coming to the HTTP server, are normal.
- N-gram model is applied in order to extract features from each request.
N-gram model

A n-gram is a sub-sequence of $n$ overlapping items (characters, letters, words, etc) from a given sequence.

Example

- `/resource?parameter1=value1&parameter2=value2.`
- `/r, re, es, so, ou, ur, . . . , lu, ue, e2.`
- `[47, 114], [114, 101], [101, 115], [115, 111], [111, 117], [117, 114], . . . , [108, 117], [117, 101], [101, 50].`
- Frequency vector of all presented subsequencies.
Self-Organizing Map

The SOM is an unsupervised, competitive learning algorithm that reduces the dimensions of data by mapping these data onto a set of units set up in much lower dimensional space.

- SOM contains a regular grid of neurones each of which is fully connected to the input layer.
- Each neuron of the SOM has an associated with it highdimensional prototype.
- At each training step, a sample vector from data set is mapped to the best matching prototype (BMU) of the SOM.
- Prototype vectors are updated so that the BMU and its topological neighbors are moved closer to the input vector in the input space.
GHSOM is a multi-layered hierarchical architecture which adapts its structure according to the input data.

- GHSOM is initialized with one SOM.
- The first SOM grows in size until it achieves an improvement in the quality of representing data.
- Each node in this map can dynamically be expanded down the hierarchy by adding a new map at a lower layer providing a further detailed representation of data.
- The procedure of growth can be repeated in new maps.
Detection method

Every new HTTP request is mapped to a GHSOM according to its resource and mapped to one of the nodes on this map by calculating the BMU for this request.

- If the distance between new request and its BMU prototype vector is greater than threshold value then this request is intrusion, otherwise it is classified as normal;
- If the node which is the BMU for the new request is classified as ”anomalous” node, then this request is intrusion, otherwise it is classified as normal.
Simulation 1: training GHSOM for one web resource

Figure: $U^*$-matrix for detecting anomalies after the training stage.
Simulation 1: GHSOM results

Table: The simulation results for different n-gram models.

<table>
<thead>
<tr>
<th>Model</th>
<th>True positive rate</th>
<th>False positive rate</th>
<th>Accuracy</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-gram</td>
<td>89.0 %</td>
<td>0.01 %</td>
<td>99.4 %</td>
<td>99.5 %</td>
</tr>
<tr>
<td>2-gram</td>
<td>99.9 %</td>
<td>0.01 %</td>
<td>99.9 %</td>
<td>99.9 %</td>
</tr>
<tr>
<td>3-gram</td>
<td>100 %</td>
<td>0.01 %</td>
<td>99.9 %</td>
<td>99.9 %</td>
</tr>
</tbody>
</table>
Since different web resources most likely use different attributes, it is reasonable to separate requests to different web resources during the preprocessing stage. Therefore, it requires the construction of an GHSOM for each unique web resource, which is not efficient from the computing resources point of view.

The process of training and the rate of classification are quite slow.
For each $k$-th unique n-gram in the training set we calculate the frequency of usage of this n-gram in the whole set $F_k^q$. Based on these frequencies we divide n-grams into two categories. This classification can be done with the help of single-linkage clustering algorithm with a predefined number of clusters equal to two.
HTTP query score

\[ S_i^q = \sum_{k \not\in \Omega} \log \left( 1 + (l_i^q - n + 1) \frac{(f_{ik}^q)^2}{F_k^q} \right), \]

where

- \( F_k^q : \sum_{k \not\in \Omega} \frac{f_{ik}^q}{F_k^q} \) is the ratio of frequency of appearance of abnormal n-grams in the \( i \)-th query and frequency of usage of those n-grams in the training set,
- \( f_{ik}^q (l_i - n + 1) \) is the number of appearance of the \( k \)-th abnormal request in the \( i \)-th query,
- the logarithmic function is used to exclude huge outliers.
- \( \Omega \) is the set of “normal“ n-grams.
DBSCAN starts with an arbitrary point that has not been visited. This point's $\epsilon$-neighborhood is found, and if it contains sufficiently many points (more than $N_{min}$), a cluster is started. Otherwise, the point is labeled as noise, although this point might later be discovered as a part of another point $\epsilon$-environment and hence be made a part of a cluster.
Classification of new HTTP requests

- $\epsilon$ is equal to absolute deviation of score values:

$$
\epsilon = \sum_{i}^{N} \left| S_{i}^{q} - \frac{1}{N} \sum_{i}^{N} S_{i}^{q} \right|.
$$

- $N_{\text{min}}$ is the maximal product of the frequency value from the cluster of abnormal $n$-grams $F_{k}^{q}$ and the total number of queries in the dataset $N^{q}$.

$$
N_{\text{min}} = \max_{k \notin \Omega} F_{k}^{q} N^{q},
$$

For new HTTP query score $S$ is calculated and the following rule is applied:

HTTP query is

$$
\begin{cases} 
\text{normal, if } S \leq \max_{i} S_{i}^{q} + \epsilon, \\
\text{intrusive, if } S > \max_{i} S_{i}^{q} + \epsilon.
\end{cases}
$$
Simulation 2: comparison with other web attacks detection techniques (one resource).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>True positive rate</th>
<th>False positive rate</th>
<th>Accuracy</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-nearest neighbor</td>
<td>59.31 %</td>
<td>0.06 %</td>
<td>97.93 %</td>
<td>97.99 %</td>
</tr>
<tr>
<td>N-gram + GHSOM</td>
<td>92.51 %</td>
<td>0.19 %</td>
<td>99.45 %</td>
<td>96.21 %</td>
</tr>
<tr>
<td>N-gram + Diffusion Maps</td>
<td>98.72 %</td>
<td>0 %</td>
<td>99.94 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Algorithm proposed</td>
<td>100 %</td>
<td>0 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>
Simulation 3: comparison with other web attacks detection techniques (several resources).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>True positive rate</th>
<th>False positive rate</th>
<th>Accuracy</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-nearest neighbor</td>
<td>55.51 %</td>
<td>2.05 %</td>
<td>97.79 %</td>
<td>59.12 %</td>
</tr>
<tr>
<td>N-gram + GHSOM</td>
<td>58.22 %</td>
<td>0.86 %</td>
<td>97.05 %</td>
<td>78.56 %</td>
</tr>
<tr>
<td>N-gram + Diffusion Maps</td>
<td>97.37 %</td>
<td>23.15 %</td>
<td>77.94 %</td>
<td>19.11 %</td>
</tr>
<tr>
<td>Algorithm proposed</td>
<td>100 %</td>
<td>0 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>
The method allows at once allows the analysis of all HTTP request messages without separating them by resource.

Simple clustering techniques are employed to find anomalous entries in the feature matrix.

The method proposed allows the detection of HTTP intrusions in case of continuously updated web-applications.

The training stage takes few minutes and the detection is fast and accurate.

In the future, we are going to design a system that is capable of detecting complex intrusions, which can consist of several HTTP requests.
Thank you for your attention!