Improving Anomaly Detection for Text-based Protocols by Exploiting Message Structures

"Security in NGNs and the Future Internet" Workshop, Berlin

Martin Güthle, Jochen Kögel, Stefan Wahl (Bell Labs), Christian Müller, Matthias Kaschub
mguethle@xunit.de

September 20th, 2010

Universität Stuttgart
Institute of Communication Networks and Computer Engineering (IKR)
Prof. Dr.-Ing. Andreas Kirstädter
Outline

Motivation

Approach

Improvement

• Extension for better detection
• Extension for higher throughput

Conclusion and Outlook
Motivation

Threat: Attacks on server

SIP: High susceptibility to vulnerabilities
- SIP server open to the outside: UNI of NGN
- SIP is complex and extensible
  - static filtering impossible
  - high probability of implementation weaknesses

Type of attacks against SIP servers
- Denial of Service
- Server integrity (e.g. gain root access) → effects thousand of millions customers
Motivation

Threat: Attacks on server

SIP: High susceptibility to vulnerabilities
- SIP server open to the outside: UNI of NGN
- SIP is complex and extensible
  - static filtering impossible
  - high probability of implementation weaknesses

Type of attacks against SIP servers
- Denial of Service
- Server integrity
Approach

Overview

Intrusion detection by anomaly detection
- Compare against model: classification
- Predefined model based on a training set

Requirements
1. Good detection rate
   - ~100% true positive
   - <0.1% false positive
2. High throughput
Approach

Feature Extraction (n-grams)

Converting text into features with numerical values

- Header fields can occur in any order
- Leverage previous work [1]
  - N-grams for feature generation
  - Dimension with good trade off between detection and performance is 4 ([1])

Principle of n-gram extraction

A sliding window is shifted over the text

```
INVITE sip:bob@exampleiNVITE.com SIP/2.0
```

extracted features:

```
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Approach

**Feature Extraction (n-grams)**

**Converting text into features with numerical values**
- Header fields can occur in any order
- Leverage previous work [1]
  - N-grams for feature generation
  - Dimension with good trade off between detection and performance is 4 ([1])

**Principle of n-gram extraction**

A sliding window is shifted over the text

```
INVITE sip:bob@exampleiNVITE.com SIP/2.0
```

Extracted features:

```
INVI
NVIT
```

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVI</td>
<td>1</td>
</tr>
<tr>
<td>NVIT</td>
<td>1</td>
</tr>
</tbody>
</table>

[1] A self-learning system for detection of anomalous SIP messages
IPTComm 2008
Approach

**Feature Extraction ( n-grams )**

Converting text into features with numerical values

- Header fields can occur in any order
- Leverage previous work [1]
  - N-grams for feature generation
  - Dimension with good trade off between detection and performance is 4 ([1])

**Principle of n-gram extraction**

A sliding window is shifted over the text

INVITE sip:bob@exampleiNVITE.com SIP/2.0

extracted features:

\[
\begin{bmatrix}
INVIT \\
NVIT \\
... \\
/2.0
\end{bmatrix}
= 
\begin{bmatrix}
1 \\
2 \\
... \\
1
\end{bmatrix}
\]

[1] A self-learning system for detection of anomalous SIP messages
IPTComm 2008
Approach

Model description and the compare unit

Classifier-based machine learning algorithm: Support Vector Machine (SVM)

- Cost factor defined with $C \in [0; \infty)$ (SVM extension [2])
- Additional extension: one class classification
- LibSVM implementation

Current limitations
- Labeled data set needed
- Training defines allowed features
- Retraining is not possible

Cost function allows outliers

Basic results

Used data set

Three different training and test data sets

- Training and test data sets are labeled
- Data sets are automatically generated, based on Codenomicon

Overview of the used data sets

<table>
<thead>
<tr>
<th>Name</th>
<th># messages</th>
<th># valids</th>
<th># invalids</th>
<th>used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train 1</td>
<td>610</td>
<td>598</td>
<td>12</td>
<td>training only</td>
</tr>
<tr>
<td>Train 2</td>
<td>928</td>
<td>900</td>
<td>28</td>
<td>training only</td>
</tr>
<tr>
<td>Test 1</td>
<td>12,923</td>
<td>2,923</td>
<td>10,000</td>
<td>test only (Train 1 + 2)</td>
</tr>
<tr>
<td>Cross</td>
<td>12,586</td>
<td>11,579</td>
<td>1,007</td>
<td>10 fold cross validation</td>
</tr>
</tbody>
</table>
Basic results

Evaluation of cost factor ($C$)

Results

• High detection rate → approach works with these sets
• Remaining problem
  – Range $\Delta C$ very narrow
  – False-Positive rate still too high
→ Improvement necessary
Basic results

What are reasons for the high False-Positive rate and narrow $\Delta C$?

- Different types of messages (Request / Response + INVITE / ACK ...)
- Optional header fields + different occurrence (e.g. multiple Via)
- Value of header fields may need session knowledge

\[
\begin{align*}
\text{SIP/2.0 180 Ringing} \\
\text{Via: SIP/2.0 ex.com;branch=abcd;} \\
\text{From: Alice <sip:alice@ex.com>} \\
\text{To: Bob <sip:bob@example.com>} \\
\text{CSeq: 1 INVITE} \\
\text{Content-Length: 0}
\end{align*}
\]

\[
\begin{align*}
\text{ACK sip:bob@example.com SIP/2.0} \\
\text{From: Alice <sip:alice@example.com>} \\
\text{To: Bob <sip:bob@example.com>} \\
\text{CSeq: 4511 ACK} \\
\text{Content-Length: 0}
\end{align*}
\]
**Improvement**

**Keyword extension**

Consider the parts which identify these reasons → **Keywords**

* A header field (e.g. Via)
* Any token inside the message (e.g. branch)

**Possible actions correspond to a keyword**

1. Keyword as additional feature
2. Replacement of session specific information
3. Start additional further processing
Improvement

Usage of the keywords

1. **Keyword as additional feature**
   
   Option 1: Occur of the keyword
   
   Option 2: Value correspond to the keyword

2. **Replace session specific information**

   ![SIP/2.0 180 Ringing](image)
   
   → Independent to the session state (comparable to noise)

3. **Start additional processing**

   These keywords call additional code (e.g. using CSeq to generate submodels)
Improvement

Evaluation with Submodels and Remove of session information

→ substantial improvement reached
## Improvement

### Throughput optimization

Influence on the throughput

- Number of features *(done)*
- Number of support vectors *(done)*
- Data structures used inside the code *(to-do)*

<table>
<thead>
<tr>
<th>Name</th>
<th>Before optimization</th>
<th>After optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train1</td>
<td>2.2 Mbps 461 msg/s</td>
<td>45.1 Mbps 9 615 msg/s</td>
</tr>
<tr>
<td>Train 2</td>
<td>3.0 Mbps 633 msg/s</td>
<td>52.4 Mbps 11 162 msg/s</td>
</tr>
<tr>
<td>Cross</td>
<td>1.4 Mbps 374 msg/s</td>
<td>14.5 Mbps 3 904 msg/s</td>
</tr>
</tbody>
</table>

![Bar chart showing throughput improvement](chart.png)
Conclusion and Outlooks

Conclusion

Anomaly detection for SIP messages based on

• Machine learning using SVM
• n-grams for feature extraction

Contribution: Significant improvement of sensitivity and detection

• Using keywords
  – As additional features
  – Removing of session information
  – Allow additional processing
• Introduction of multiple models

Throughput optimization

Outlook

• Definition of the training traces
• Simplify the expendability to any kind of SIP extensions
• Extend the detection method to other text based protocols