User behaviour analysis for Malware detection

Valentina Dumitrasc & René Serral
Introduction
Increase in cyberattacks
Malware attacks increased 358% compared to 2019
68% of the companies experienced a targeted attack on their networks

Current techniques
Current techniques for malware detection are limited
**Dynamic analysis**
- Detected by malware
- Change behaviour
- Computer resources needed
- Not immediate
- Not detect until executed
- No specific solution for servers
- Malware-centric

**Static analysis**
- Malware signatures change
- Packed/Obfuscated
- Can’t detect zero days
- No specific solution for servers
- Malware-centric
Detect deviation from normal user behaviour

User-centric

Does not involve a dedicated infrastructure for malware analysis

Efficient

Metrics extracted every few seconds

Quick detection

Detect any deviation

Zero-days

Adapt to any host

Servers

Models are quickly trained + fast prediction

Lightweight
Architecture

Part II
Architecture

Wazuh Agent + Custom Scripts

Elastic Search

Custom AI/ML Engine

Enduser host
- System Stats
- Network Stats
- Logs

Endpoint Agent

SBA Engine
- Long Term Storage
- Feature Extraction
- Data Adaptation
- Data Broker
- Decision Engine
Relationships among features
Can handle noise
Adapt to changes

We propose a system able to adapt without previous supervised learning process
Part III

Model & Scenario

Part III
**Network**
- Geographical connection origin
- Data Traffic Ratio
- Data Transfer Rate
- Packet Ratio
- Packet Transfer Rate
- Connection Status/Count

26

**Processes**
- Process count

3

**CPU/Memory**
- Cpu usage
- Memory usage

15

**System audit**
- File Operations
- Command ratio

5
Normal behaviour data

- Audio/Video Streaming
- Online text editing
- Online model training and testing
- Command execution
- Information searching
- Cloud based storage
- Idle state
Test data

- Ransomware
- Denial of Service
- Botnet
- Backup
- Software compilation
- Normal behaviour
Algorithms

- One-Class SVM
- Local Outlier Factor
- Kernel Density Estimation
- Autoencoder
Kernel Density Estimation

Parameters

kernel $\rightarrow$ Gaussian
bandwidth $\rightarrow$ Scott
threshold $\rightarrow$ 7th prc.

Issues

Wrong feature importance
Issues detecting botnet and ransomware
Autoencoders

**Layers**

- **input**: Shape of the training data
- **encoding**: $49 \times n$, ReLU activation, L1 regularization
- **decoding**: $49 \times n$ ReLU
- **output**: sigmoid function

**Issues**

- Poor feature importance
- Issues detecting ransomware
Kernel Density Estimation
Autoencoder

- Ransomware: 100%
- Botnet: 71%
- Denial of Service: 100%
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>One-Class SVM</th>
<th>LOF</th>
<th>Autoencoder</th>
<th>KDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal behaviour</td>
<td>0.9</td>
<td>0.92</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Botnet</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>DoS</td>
<td>1</td>
<td>0.96</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ransomware</td>
<td>1</td>
<td>0.5</td>
<td>0.71</td>
<td>0.42</td>
</tr>
</tbody>
</table>
Further training the system:
Introducing software compilation and backup data

<table>
<thead>
<tr>
<th></th>
<th>One-Class SVM</th>
<th>LOF</th>
<th>Autoencoder</th>
<th>KDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal behaviour</td>
<td>0.81</td>
<td>0.95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Botnet</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DoS</td>
<td>1</td>
<td>0.96</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ransomware</td>
<td>0.93</td>
<td>0.28</td>
<td>0.78</td>
<td>0.35</td>
</tr>
</tbody>
</table>
Next Steps / Conclusions

Part V
Next Steps

Real life detection/mitigation

Test/Incorporate into a real system

Cloud

Preprocessing

Different operating systems
THANKS

QUESTIONS?