Host-Based Anomaly Detection

Wahab Hamou-Lhadj, PhD, ing.
Software Behaviour Analysis (SBA) Research Lab
ECE, Concordia University
Montreal, QC, Canada
wahab.hamou-lhadj@concordia.ca

Feb. 6, 2014
Ottawa, ON, Canada
Objectives

- Protect **host systems against cyber-attacks** (web-based exploitation, simulated social engineering, etc.)

- Model **system health** and develop **modular, adaptive, and scalable** Anomaly Detection Systems (ADS) at the **system call level**

- Reduce **false positives (alarms)** and improve the **true positives**

- Provide preliminary analysis/recommendations for future research and directions
Background on ADS

- Monitors computer or network activity for signs of intrusions and alert administrators
- Signature based Detection
  - Looks for known patterns
  - Detects only known attacks
- Anomaly Detection
  - Looks for deviations from normal behavior
  - Detects even unknown attacks (zero day exploits)
Analyze, correlate, look for trends, etc.

Run-Time Information (traces, profiling data)

Build a Reference Model (normal behaviour)

Model of The System

Analyze, correlate, look for trends, etc.

Deviation from normal?

Yes

Decide what to do (automatically and/or user-guided)

Need for repair?

Yes

initiate repair actions

No

continue normal execution

continue normal execution

In-Lab

In-Ops

System
Analyze, correlate, look for trends, etc.

Run-Time Information (traces, profiling data)

Model of The System

Build a Reference Model (normal behaviour)

System

Deviation from normal?

continue normal execution

initiate repair actions

Need for repair?

continue normal execution

in-Lab

In-Ops

Decide what to do (automatically and/or user-guided)
Existing Work

Several techniques have been used to model the normal behavior of a system

- Sliding window technique
- HMM
- Neural networks (two-class)
- Clustering
- Varied length n-gram technique
- Context Free Grammar
Example: Sliding Approach (STIDE)

Model composed of sequences of k size

open, read, mmap, mmap, open, read, read, ...

Correlation Algorithm
(number/percentage of mismatches, Hamming distance)

Decision?
Challenges – False alarms

- High false alarms **reduce confidence** and could lead to deactivation of the ADS

- Causes:
  - Unrepresentative normal data for training and attack data for validation and testing
  - Inappropriate model or feature selection
  - Poor optimization of models parameters
  - Over fitting (leads to poor generalization)
  - Inadequate assumptions such as static environments
Challenges: Adaptability

- ADSs are often designed using limited data
  - collection and analysis of representative data from each process (different versions, OS, etc.) is costly

Anomaly detector will have **incomplete** view of normal system behavior
In Practice

• Dynamic environment
  – Changes in normal process behavior due, for instance, to application update

Internal model of normal behavior **diverges** with respect to the underlying data
ADS Requirements

ADS should:

- Account for rare normal events (false alarms)
- Be scalable and modular: can add, replace or remove models or features over time
- Handle large data spaces
- Accommodate new data
Advanced Host-Level Surveillance

AHLS Track 3

- Attack Taxonomy and Manifestation
  - Dataset Generation
  - Evaluation Protocol
- Data Preparation
- Data Space Reduction
  - Model Selection
  - Feature Selection
- Data Abstraction
- Incremental Learning
  - Model Combination
  - Online Learning
- Adaptive Learning
- Anomaly Detection System Integration
  - Integration of Human Feedback
  - Prototyping
  - Infrastructure
Advanced Host-Level Surveillance

AHLS Track 3

- Attack Taxonomy and Manifestation
- Dataset Generation
- Evaluation Protocol
- Data Preparation

- Data Space Reduction
- Model Selection
- Feature Selection
- Data Abstraction

- Incremental Learning
- Model Combination
- Online Learning
- Adaptive Learning

- Anomaly Detection System Integration
- Integration of Human Feedback
- Prototyping
- Infrastructure
Kernel State Modeling (KSM)

• KSM is an anomaly detection technique
  – Transforms system calls into kernel modules, called states
  – Detect anomalies at the level of interaction of kernel states
  – Reduces data space used in training and testing
  – Favors efficiency while keeping accuracy
Transforming System Calls into States of Kernel Modules

<table>
<thead>
<tr>
<th>State</th>
<th>Module in Linux Source Code</th>
<th># of System Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Architecture</td>
<td>10</td>
</tr>
<tr>
<td>FS</td>
<td>File System</td>
<td>131</td>
</tr>
<tr>
<td>IPC</td>
<td>Inter Process Communication</td>
<td>7</td>
</tr>
<tr>
<td>KL</td>
<td>Kernel</td>
<td>127</td>
</tr>
<tr>
<td>MM</td>
<td>Memory Management</td>
<td>21</td>
</tr>
<tr>
<td>NT</td>
<td>Networking</td>
<td>2</td>
</tr>
<tr>
<td>SC</td>
<td>Security</td>
<td>3</td>
</tr>
<tr>
<td>UN</td>
<td>Unknown</td>
<td>37</td>
</tr>
</tbody>
</table>

[Source]: http://syscalls.kernelgork.com
KSM and Density Plots

Density Plot

Freq. (n=16800)


FS MM AC
KL NT
Anomaly Detection in Firefox

Normal

Anomalous
Anomaly Detection in Login Utility

Normal

Anomalous
Automatically Detecting Anomalies

• To determine significant deviation threshold (alpha):
  – Divide normal dataset into training set, validation set, and testing set
  – Extract probabilities from training set
  – Evaluate on validation set and adjust alpha
  – Measure accuracy on testing set
Case Study 1: ADFA Linux Dataset

- A host with Ubuntu 11.04, Apache 2.2.17, PHP 5.3.5, TikiWiki 8.1, FTP server, MySQL 14.14 and an SSH server
  - web-based exploitation
  - simulated social engineering
  - poisoned executable,
  - remotely triggered vulnerabilities,
  - remote password brute force attacks
  - system manipulation
## Case Study 1: ADFA Linux Dataset

<table>
<thead>
<tr>
<th>Set</th>
<th># of training traces</th>
<th># of attacks</th>
<th># of normal traces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td></td>
<td>833</td>
<td></td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td># of attacks</td>
<td>20</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td># of attacks</td>
<td>40</td>
<td>3373</td>
</tr>
</tbody>
</table>
Receiver Operating Characteristics (ROC) Curves

- **True Positive:** anomaly detected as anomaly
- **False Positive:** normal detected as anomaly
Case Study 1: ADFA Linux Dataset
## Case Study 2: Dataset

<table>
<thead>
<tr>
<th>Program</th>
<th># Normal Traces</th>
<th>#Attack Types</th>
<th>#Attack Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Normal Traces</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Training</td>
<td>Validation</td>
<td>Testing</td>
</tr>
<tr>
<td>Login</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>PS</td>
<td>10</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Stide</td>
<td>400</td>
<td>200</td>
<td>13126</td>
</tr>
<tr>
<td>Xlock</td>
<td>91</td>
<td>30</td>
<td>1610</td>
</tr>
<tr>
<td>Firefox</td>
<td>125</td>
<td>75</td>
<td>500</td>
</tr>
</tbody>
</table>
## Case Study 2: Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Technique</th>
<th>TP rate</th>
<th>FP rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>KSM (alpha=0.00)</td>
<td>100%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=6)</td>
<td>100%</td>
<td>40.00%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=10)</td>
<td>100%</td>
<td>40.00%</td>
</tr>
<tr>
<td></td>
<td>HMM (states=10)</td>
<td>100%</td>
<td>40.00%</td>
</tr>
<tr>
<td>PS</td>
<td>KSM (alpha=0.02)</td>
<td>100%</td>
<td>10.00%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=6)</td>
<td>100%</td>
<td>10.00%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=10)</td>
<td>100%</td>
<td>10.00%</td>
</tr>
<tr>
<td></td>
<td>HMM (states=5)</td>
<td>100%</td>
<td>30.00%</td>
</tr>
<tr>
<td>Xlock</td>
<td>KSM (alpha=0.04)</td>
<td>100%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=6)</td>
<td>100%</td>
<td>1.50%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=10)</td>
<td>100%</td>
<td>1.50%</td>
</tr>
<tr>
<td></td>
<td>HMM (states=5)</td>
<td>100%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
## Case Study 2: Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Technique</th>
<th>TP rate</th>
<th>FP rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stide</td>
<td>KSM (alpha=0.06)</td>
<td>100%</td>
<td>0.25%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=6)</td>
<td>100%</td>
<td>4.97%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=10)</td>
<td>100%</td>
<td>5.25%</td>
</tr>
<tr>
<td></td>
<td>HMM (states=5)</td>
<td>100%</td>
<td>0.25%</td>
</tr>
<tr>
<td>Firefox</td>
<td>KSM (alpha=0.08)</td>
<td>100%</td>
<td>0.60%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=6)</td>
<td>100%</td>
<td>44.60%</td>
</tr>
<tr>
<td></td>
<td>Stide (win=10)</td>
<td>100%</td>
<td>49.20%</td>
</tr>
<tr>
<td></td>
<td>HMM (states=5)</td>
<td>100%</td>
<td>1.40%</td>
</tr>
</tbody>
</table>
## Case Study 2: Execution Time

<table>
<thead>
<tr>
<th></th>
<th>Size of All Traces</th>
<th>KSM</th>
<th>Stide</th>
<th>HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>26.2KB</td>
<td>4.46 sec</td>
<td>0.03 sec</td>
<td>56.43 min</td>
</tr>
<tr>
<td>PS</td>
<td>29.6KB</td>
<td>5.14 sec</td>
<td>0.11 sec</td>
<td>46.24 min</td>
</tr>
<tr>
<td>Xlock</td>
<td>47.4MB</td>
<td>1.51 min</td>
<td>12.3 min</td>
<td>13.37 hr</td>
</tr>
<tr>
<td>Stide</td>
<td>36.2MB</td>
<td>5.85 min</td>
<td>8.53 min</td>
<td>2.3 day</td>
</tr>
<tr>
<td>Firefox</td>
<td>270.6MB</td>
<td>9.35 min</td>
<td>4.17 hr</td>
<td>4.03 day</td>
</tr>
</tbody>
</table>
Research Threads

AHLS Track 3

- Attack Taxonomy and Manifestation
  - Dataset Generation
  - Evaluation Protocol
- Data Preparation
- Data Space Reduction
  - Model Selection
  - Feature Selection
- Data Abstraction
- Incremental Learning
  - Model Combination
  - Online Learning
- Adaptive Learning
- Anomaly Detection System Integration
  - Integration of Human Feedback
  - Prototyping
- Infrastructure
Model Combination

- A single classifier or model may not provide a good approximation to the underlying data structure or distribution
  - No dominant classifier for all data distributions ("no free lunch" theorem)
  - True data distribution is usually unknown
  - Limited amount of (labeled) data is typically provided during training
IBC: Iterative Boolean Combination in the ROC Space

- For each threshold from the first detector and each threshold from the second detector:
  - Combine the responses using all Boolean functions
  - Select thresholds and Boolean functions that improve the ROC space
IBC - Example
## Experimental Methodology

<table>
<thead>
<tr>
<th>Set</th>
<th># of training traces</th>
<th># of attacks</th>
<th># of normal traces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training Set</strong></td>
<td>833</td>
<td>20</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Validation Set</strong></td>
<td></td>
<td>20</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Testing Set</strong></td>
<td>40</td>
<td></td>
<td>3373</td>
</tr>
</tbody>
</table>
Combination of Responses from Different HMMs

HMM (N=20)  
HMM (N=200)  
IBC  
Anomaly  
Normal
Combination Results on Validation Set

True positive rate

False alarm rate

- $M_1$: HMM($N=200$), auc=0.933
- $M_2$: HMM($N=20$), auc=0.845
- IBC($M_1$, $M_2$), auc=0.986
Combination Results on Test Set

True positive rate vs. False alarm rate

- $M_1$: HMM(N=200), auc=0.919
- $M_2$: HMM(N=20), auc=0.818
- IBC($M_1$, $M_2$), auc=0.977
- Creech & Hu 2013, auc=0.954
Combination of HMM and STIDE Responses

HMM (N=200) -> IBC

STIDE (WS = 5) -> IBC

IBC

Anomaly

Normal
Combination Results on Validation Set

True positive rate vs. False alarm rate

- $M_1$: HMM($N=200$), $auc=0.933$
- $M_2$: STIDE($WS=5$), $auc=0.965$
- IBC($M_1$, $M_2$), $auc=0.992$
TotalADS

- TotalADS is an integrated Anomaly Detection System Environment
  - Eclipse Plug-in
  - Open Source
  - Based on TMF (Tracing and Monitoring Framework)
  - Supports STIDE, HMM, KSM, IBC
  - Supports a combination of classifiers
  - Supports trace analysis and forensic analysis
  - Supports CTF (Common Trace Format)
Architecture

Trace Mgt
- Controller
- Loading
- Streaming
- CTF

Training/Detection Engine
- HMM
- STIDE
- KSM
- IBC
- Others

UI
- Plots
- Controllers
- Statistics
- Reports
- Analysis

DBMS
Anomaly detection for zero day attacks

Trace (system calls)
Training models
Conclusion

- **Research threads**: Data preparation, data abstraction, adaptive learning, and infrastructure
- **ADS requirements**: Low false positive rate, scalability, and adaptability
- **KSM**: Abstraction is not the enemy of accuracy
- **IBC**: Combining detectors provides better results than using a single detector
- **TotalADS**: An environment for integrating multiple anomaly detection systems
Future Plans

- Continue experimenting with KSM and IBC on other datasets (preferably generated at DRDC)
- Combine additional detectors using IBC
- Start working on adaptive/incremental learning
- Continue improving the maturity level of TotalADS
- Integrate this work with work done at other universities
- Transfer knowledge to DRDC
Thank You

Wahab Hamou-Lhadj, PhD, ing.
Software Behaviour Analysis (SBA) Research Lab
Concordia University
Montreal, QC, Canada

www.ece.concordia.ca/~abdelw/sba