IDS Using Machine Learning Techniques

COMP 290-40
Brian Begnoche
March 23, 2005

Overview

• What is ML?
• Why use ML with IDS?
• Host-based ML methods
  ♦ 3 examples
• Network-based ML methods
  ♦ 2 examples
• Using ML to improve existing NIDSs
  ♦ 2 examples

Why ML?

• Find patterns of malicious activity
  ♦ difficult and tedious
  ♦ attacks are complex, spatially and temporally
  ♦ stealthy “low and slow” attacks
  ♦ Behavior-based, rather than knowledge-based
• Automation
  ♦ automatically generate rules from training set
  ♦ complete automation not always desirable
  ♦ decision aids for the sys admin

What is Machine Learning?

• Allow computers to “learn”
• Supervised learning
  ♦ Program learns how to behave from predetermined data set
• Unsupervised learning
  ♦ Program learns as it receives input, improving over time
• Collaborative approach between human and machine

ML Techniques

• Host-based
  ♦ Time-based Inductive Learning (1990)
  ♦ ML anomaly detection (1997)
  ♦ Instance-Based Learning (1999)
• Network-based
  ♦ Network Exploitation Detection Analyst Assistant (1999)
    ♦ Genetic algorithms and decision trees
  ♦ Portscan Detection (2004)
    ♦ Threshold Random Walk
• Real-time anomaly detection
  ♦ Unusual or unrecognized activities
• Sequential rules based on user’s behavior over time
  ♦ UNIX commands
• Checked with rulebase
  ♦ Static approach: site security policy
  ♦ Dynamic approach: time-based inductive machine (TIM)
**Time-based Inductive Machine (TIM)**

- Discovers temporal patterns of highly repetitive activities
  - Patterns described by rules
- Rules generated/modified by inductive generalization
- Input to TIM is an *episode*
  - *Episode = sequence of events*

**Example TIM rules**

- **E1 - E2 - E3 --> (E4 = 95%; E5 = 5%)**
  - Sequence of events E1, E2, E3
  - Next event E4 95% of the time, E5 the other 5%
- **A-B-C-S-T-S-T-A-B-C-A-B-C**
  - R1: A-B --> (C, 100%)
  - R2: C --> (S, 50%; A 50%)
  - R3: S --> (T, 100%)
  - R4: T --> (A, 50%; S, 50%)
Similarity Measure

• Sim(Seq₁, Seq₂):
  ♦ Algorithm
    • Adjacency counter $c := 1$
    • Similarity measure $Sim := 0$
    • For each position $i$ in sequence length
      - If Seq₁($i$) = Seq₂($i$) then $Sim := Sim + c$ and increment $c$
      - Otherwise, $c := 1$
  ♦ Bounded by $n(n+1)/2$, $n=$seq. Length
  ♦ Biased toward adjacent identical tokens
  ♦ Similarity to dictionary is similarity to most similar sequence in dictionary

Similarity Measure Example

```
cd <l> ls -lArF tar <2> less <2>
```

```
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Final Similarity Score: 9

Smoothed Similarity

• Windowed mean-value filter

$$m_s(i, L) = \frac{1}{w} \sum_{j=i-w}^{i} Sim(Seq_j, L)$$

Testing Differentiation

• 4 users' UNIX command histories
  ♦ Seq. length = 12, dictionary size = 2000
  ♦ Each user tested against all user profiles
  ♦ Should result in high “sameness” when compared with itself
    ♦ Where are true positives? False?

<table>
<thead>
<tr>
<th>Profiled User</th>
<th>Tested User</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER₀</td>
<td>USER₁ USER₂ USER₃</td>
</tr>
<tr>
<td>USER₁</td>
<td>17.64 28.30 23.32 1.25</td>
</tr>
<tr>
<td>USER₂</td>
<td>3.52 54.66 72.10 8.29</td>
</tr>
<tr>
<td>USER₃</td>
<td>6.27 15.74 11.52 69.80</td>
</tr>
</tbody>
</table>

Unit = % of windows labeled as same user

ML Techniques

• Host-based
  ♦ Time-based Inductive Learning (1990)
  ♦ ML anomaly detection (1997)
  ♦ Instance-Based Learning (1999)

• Network-based
  ♦ Network Exploitation Detection Analyst Assistant (1999)
    ♦ Genetic algorithms and decision trees
  ♦ Portscan Detection (2004)
    ♦ Threshold Random Walk

Instance-Based Learning

• Cyclic process
  ♦ Compare sequences with user profile
  ♦ Filter out noise from similarity measure
  ♦ Classify sequence by threshold decision
  ♦ Feedback classification to adjust profile over time
**IBL Flow**

Fig. 1. Information flow in the instance-based anomaly-detection system.

**IBL Accuracy**

- Similar test as before
  - All users tested against user 6
  - % of sequences correctly identified
  - +: true negative
  - o: true positive

**IBL Time-to-Alarm**

- Time measured in token count
- +: true positive
  - Rapid detection
- o: false positive
  - Slower detection
  - Clustered

**IBL Storage Reduction**

- Instance selection
  - Prediction: Recent sequences will be used again
  - Limit profile size by selection
    - FI FO, LRU, LFU, random
    - FI FO worst
  - LRU and LFU performed best
    - Lose ~3.6% accuracy on true accept rate
    - Gain ~3.5% accuracy on true detect rate
  - False positives? Paper didn’t say...
  - All methods improved time-to-alarm

**Selection Comparison**

Instance-based accuracy vs normal accuracy
- +: true detect
- o: true accept

**Selection Time-to-Alarm**

Instance-based TTA vs normal TTA
- +: true alarms
- o: false alarms
IBL Storage Reduction

- Instance clustering
  - Use distance measure to cluster nearby points
  - \( \text{Dist}(X,Y) = \text{Sim}(X,X) - \text{Sim}(X,Y) \)
  - Two approaches:
    - \( K \)-centers: predetermined number of clusters \( K \)
    - Greedy clustering: add points to cluster until mean intercluster distance \( \text{val}(C) \) drops below a threshold \( C \)

Comparing Cluster Methods

- Use distance measure to cluster nearby points
- Two approaches:
  - \( K \)-centers: predetermined number of clusters \( K \)
  - Greedy clustering: add points to cluster until mean intercluster distance \( \text{val}(C) \) drops below a threshold \( C \)

ML Techniques

- Host-based
  - Time-based Inductive Learning (1990)
  - ML anomaly detection (1997)
  - Instance-Based Learning (1999)
- Network-based
  - Network Exploitation Detection Analyst Assistant (1999)
    - Genetic algorithms and decision trees
  - Portscan Detection (2004)
    - Threshold Random Walk

Network Exploitation Detection Analyst Assistant (NEDAA)

- Automatically generate rules for classifying network connections
  - Normal or anomalous
- Two independent, parallel ML methods to generate rules
  - Genetic algorithms
  - Decision trees
- Basically a proposal, paper has no results

Genetic Algorithms

- Based on evolution and natural selection
- Find optimal solutions
  - Potential solution = gene
  - Coded sequence of solution = chromosome
  - Set of genes = population
- “Fitness” of a gene
  - Rule used to filter marked dataset
  - Rewarded for full/partial matches of anomalies, penalized for normal matches
- Two ways that genes evolve
  - Reproduction: New gene created from existing genes
  - Mutation: Gene randomly changes
- Chromosome survival and recombination is biased toward fittest genes
- After certain number of generations, best rules selected
**Example Chromosome**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP</td>
<td>42.22.a5.b.c (66.34.229.188)</td>
</tr>
<tr>
<td>Dest IP</td>
<td>15.6.e.76 (21.1.76 + 7.110.118)</td>
</tr>
<tr>
<td>Source port</td>
<td>047051</td>
</tr>
<tr>
<td>Dest port</td>
<td>912320</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP</td>
</tr>
</tbody>
</table>

- **Chromosome:**
  - $(4,2,2,14,5,11,12,1,5,11, -1,6,14,7,6,0,4,7,0,5,1,9,1,2,3,2,0,17)$

---

**Decision Trees**

- Classify data with common attributes
  - Remember snort's decision tree?
- Each node specifies an attribute
  - Each leaf is a decision value
    - i.e. Normal or anomalous
- Paper uses ID3 algorithm
  - Use training set to construct tree
  - Prune tree to normal only

---

**ML Techniques**

- Host-based
  - Time-based Inductive Learning (1990)
  - ML anomaly detection (1997)
  - Instance-Based Learning (1999)
- Network-based
  - Network Exploitation Detection Analyst Assistant (1999)
    - Genetic algorithms and decision trees
  - Portscan Detection (2004)
    - Threshold Random Walk

---

**Portscan detection**

- Identify malicious portscanners
  - Hosts are either benign or a scanner
- Major goal: balance promptness and accuracy
- Threshold Random Walk (TRW)
  - Online detection algorithm to detect scanners
  - Uses Sequential Hypothesis Testing

---

**Sequential Hypothesis Testing**

- Uses idea that a successful connection attempt is more likely to come from a benign host
- Choose a hypothesis based on a series of events
  - $H_0$: host is benign
  - $H_1$: host is a scanner
  - Event $Y_1 = 0$ if a connection attempt by host is a success, 1 if a failure
Choosing a Hypothesis

- Observe events until one of two thresholds met
  - \( \Lambda(Y) = \frac{\Pr[Y|H_1]}{\Pr[Y|H_0]} \)
  - \( \frac{\Pr[Y|H_k]}{\Pr[Y|H_k]} = \frac{\Pr[Y|H_0]}{\Pr[Y|H_0]} \)

Figure 3. Flow diagram of the real-time detection algorithm

Evaluating TRW

- Three measures
  - Efficiency: ratio of true positives to total number of hosts flagged as scanners
  - Effectiveness: ratio of true positives to all scanners (detection rate)
  - Number of connections required to decide on a hypothesis

Pros of TRW

- Compared with snort and bro
- Improved effectiveness
- Faster detection (N)

<table>
<thead>
<tr>
<th>Measures</th>
<th>TRW</th>
<th>Bro</th>
<th>Snort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>0.960</td>
<td>1.000</td>
<td>0.615</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.960</td>
<td>0.150</td>
<td>0.126</td>
</tr>
<tr>
<td>N</td>
<td>6.08</td>
<td>21.40</td>
<td>14.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures</th>
<th>TRW</th>
<th>Bro</th>
<th>Snort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.990</td>
<td>0.029</td>
<td>0.323</td>
</tr>
<tr>
<td>N</td>
<td>6.08</td>
<td>24.91</td>
<td>6.08</td>
</tr>
</tbody>
</table>

Cons of TRW

- Easy to camouflage a scan
  - Intermingle valid connection attempts with scan attempts
- Web spiders look like scanners
- Proxies can get flagged as scanner rather than source
- DoS as result of address spoofing
  - Act like a scanner, spoofing address, so that target’s real traffic also gets dropped

Improving NIDSs

- KDD 1999 CUP dataset
  - KDD Cup is the annual Data Mining and Knowledge Discovery competition
  - 1999 evaluated various NIDS methods
  - Contained four major attack categories
- Data mining NIDS alarms
  - Handle alarms more efficiently

KDD 1999 CUP dataset

- Tested nine ML methods for NIDS
- Two datasets
  - Labeled dataset: training
  - Unlabeled dataset: testing
- Covers four major attack categories
  - Probing: information gathering
  - DoS
  - User-to-root (U2R): unauthorized root access
  - Remote-to-local (R2L): unauthorized local access from remote machine
The nine KDD Cup methods

- Multilayer perceptron (MLP)
- Gaussian classifier (GAU)
- K-means clustering (K-M)
- Nearest cluster algorithm (NEA)
- Incremental radial basis function (IRBF)
- Leader algorithm (LEA)
- Hypersphere algorithm (HYP)
- Fuzzy ARTMAP (ART)
- C4.5 Decision tree (C4.5)

KDD Cup Results

- Probability of detection and false alarm rate
- No method won
- Some methods better for different attacks
- Conclusion? Use multiple methods!

Data mining NIDS alarms

- Learn how to handle future alarms more efficiently
  - Partial automation
  - Manual investigation of alarms is labor-intensive and error-prone
  - Up to 99% of alarms are false positives
- Two different techniques
  - Episode rules
  - Conceptual clustering

Episode Rules

- Predict the occurrence of certain alarms based on occurrence of other alarms
  - Ex.: 50% of “Auth. Failure” alarms followed within 30s by “Guest Login” alarm
- Episode rule form
  - \(<P_1, ..., P_k> = <P_1, ..., P_k, ..., P_n> [s, c, W]\)
  - RHS has minimum \(s\) occurrences in sequence \(S\)
  - RHS occur within time \(W\) after LHS with confidence \(c\)

Results from Episode Rules

- Characteristic episodes of attack tools
- RHS represented massive attack, LHS was early indicator of attack
- Some alarms almost always entail other alarms
  - Ex.: “TCP FIN Host Sweep” implies “Orphaned FIN Packet”
- Discovered legitimate episodes

Episode Rule Drawbacks

- Attainable degree of automation very low
  - <1% of alarms could be handled automatically based on previous episodes
- Tends to produce large number of irrelevant/redundant patterns
- Many patterns difficult to interpret
Conceptual Clustering

- Group events into categories
- Try to use abstract values
  - IP address => network
  - Timestamp => weekday
  - Port number => port range
- Generalization hierarchy
  - Is-a relationship
- Careful not to over-generalize from noise

Generalization Hierarchy

<table>
<thead>
<tr>
<th>SrcIP</th>
<th>DstIP</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip1</td>
<td>ip4</td>
<td>1000</td>
</tr>
<tr>
<td>ip1</td>
<td>ipA1</td>
<td>1</td>
</tr>
<tr>
<td>ip7</td>
<td>ipB1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ipA1</td>
<td>ipZ1</td>
<td>1</td>
</tr>
<tr>
<td>ipB1</td>
<td>ip4</td>
<td>1</td>
</tr>
<tr>
<td>ipZ1</td>
<td>ip4</td>
<td>1</td>
</tr>
</tbody>
</table>

a) Generalization hierarchy for IP addresses.

b) Sample table.

Figure 3: A generalization hierarchy and sample table.

Summary

- **ML to improve IDS**
  - Automation
  - Efficiency
  - Ease of use
  - Make sense of alarms