# 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**



## 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**



## 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**



## 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



## Time-based Inductive Learning

- **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%)**



## Inductive Generalization

- **Update rules until rulebase consists of *high quality* hypotheses**
  - ♦ **High accuracy in prediction**
    - **Hypothesis is correct most of the time**
    - **Described as *entropy***
    - **Entropy =** $\Sigma_i(-p_i log(p_i))$
  - ♦ **High level of confidence**
    - **Hypothesis confirmed by many observations**



## 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



## ML Anomaly Detection

- **Compare command sequences w/ user profile**
  - ♦ **behavior, not content**
  - ♦ **HCI is causal**
  - ♦ **Empirically, best length 8-12 commands**
- **Based on positive examples of valid user behavior**
- **Similarity measure**



## Example command sequence

- **Example command stream:**
  - ♦ **> ls -laF**
    **> cd /tmp**
    **> gunzip -c foo.tar.gz | (cd \ ; tar xf -)**
- **Translated into token stream:**
  - ♦ **ls -laF cd <1> gunzip -c <1> | ( cd <1> ; tar - <1> )**

## Similarity Measure

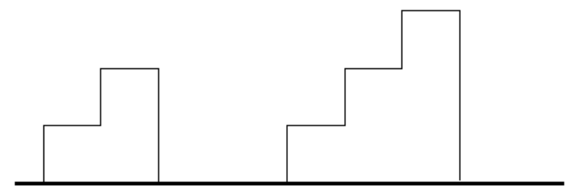
- $\mathrm{Sim}(\mathrm{Seq}_1, \mathrm{Seq}_2)$:
  - ♦ **Algorithm**
    - Adjancy counter *c := 1*
    - *Similarity measure Sim := 0*
    - *For each position i in sequence length*
      - *If $Seq_1(i) = Seq_2(i)$ then Sim := Sim + c and increment c*
      - *Otherwise, c := 1*
  - ♦ ***Bounded by 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 <1> ls -laF tar <2> less <2>

cd <1> cat <1> tar <2> less <1>
```



Final Similarity Score: 9



## Smoothed Similarity

- **Windowed mean-value filter**

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

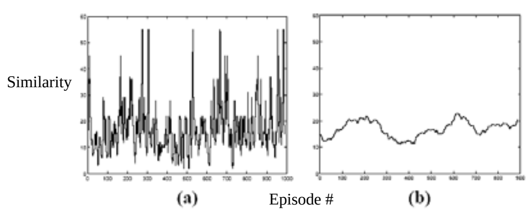



Figure 1: Similarity measure stream. (a) Raw. (b) Smoothed.



## 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?**

| Profiled User | Tested User | | | |
|---|---|---|---|---|
| | USER0 | USER1 | USER2 | USER3 |
| USER0 | 99.19 | 35.35 | 6.11 | 0.00 |
| USER1 | 17.84 | 88.30 | 23.32 | 1.25 |
| USER2 | 3.52 | 54.86 | 72.10 | 8.29 |
| USER3 | 6.27 | 15.74 | 11.52 | 69.85 |

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 adust profile over time**

## IBL Flow

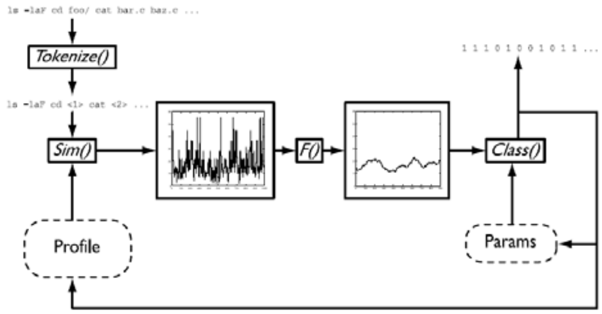
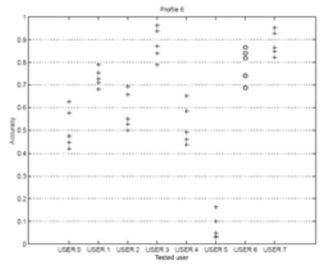



Fig. 1. Information flow in the instance-based anomaly-detection system.



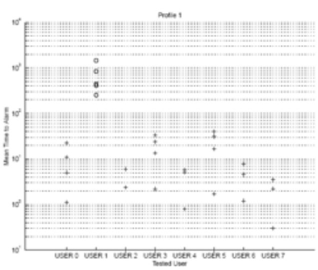
## IBL Accuracy



- **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

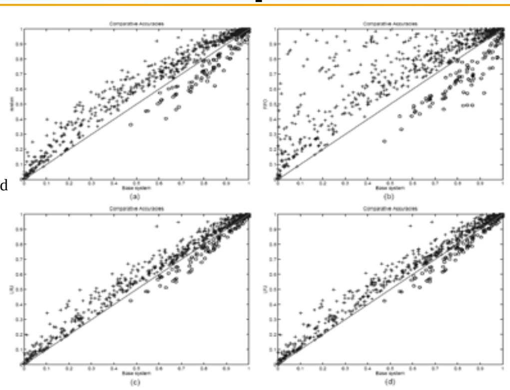
- **Instance selection**
  - **Prediction: Recent sequences will be used again**
  - **Limit profile size by selection**
    - **FIFO, LRU, LFU, random**
  - **FIFO worst**
  - **LRU and LFU performed best**
    - **Lose ~3.6% accuracy on true accept rate**
    - **Gain ~3.5% accuracy on true detect 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

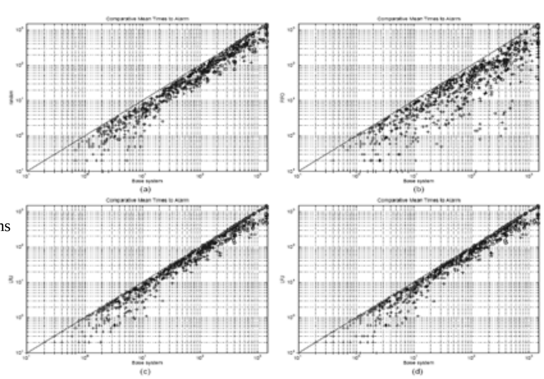




## Selection Time-to-Alarm

Instance-based TTA vs normal TTA

+: true alarms
o: false alarms



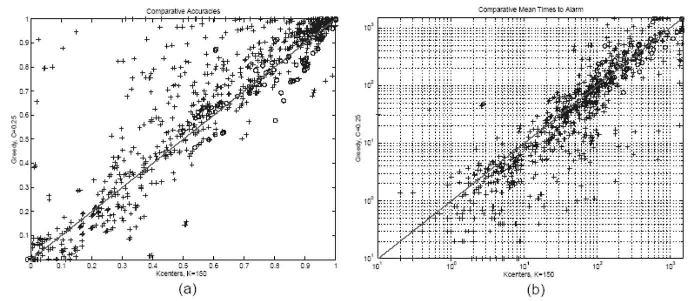
## IBL Storage Reduction

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

$$val(\mathbf{C}) = \frac{\sum_{x \in \mathbf{C}} \sum_{y \in \mathbf{C}} \mathrm{Dist}(x, y)}{|\mathbf{C}|^2}$$



## Comparing Cluster Methods




Insignificant difference in accuracy, but greedy clustering has better TTA



## 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**



## Genetic Algorithms

- **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

| Attribute | Value |
|---|---|
| Source IP | 42.22.e5.bc(66.34.229.188) |
| Dest IP | 15.b*.6e.76(21.176+?.110.118) |
| Source port | 047051 |
| Dest port | 912320 |
| Protocol | TCP |

- **Chromosome:**
  - **(4,2,2,2,14,5,11,12,1,5,11, -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**



## Decision Tree Example

**Table 1. Example Intrusion Data**

| *IP Port* | *System Name* | *category* |
|---|---|---|
| 004020 | Artemis | normal |
| 004020 | Apollo | intrusion |
| 002210 | Artemis | normal |
| 002210 | Apollo | intrusion |
| 000010 | Artemis | normal |
| 000010 | Apollo | normal |

**Figure 1. Example Intrusion Decision Tree**

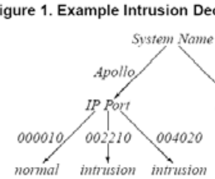
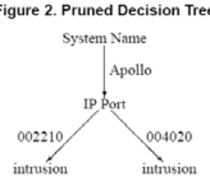



**Figure 2. Pruned Decision Tree**






## 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 $Y_i$ = 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
  - $\Lambda(Y) = Pr[Y|H_1]/Pr[Y|H_0]$
  - $Pr[Y|H_k] = \Pi\, Pr[Y_i|H_k]$

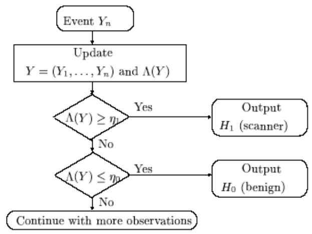



Figure 3. Flow diagram of the real-time detection algorithm



## 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)

| | Measures | TRW | Bro | Snort |
|---|---|---|---|---|
| LBL | Efficiency | 0.963 | 1.000 | 0.615 |
| | Effectiveness | 0.960 | 0.150 | 0.126 |
| | N | 4.08 | 21.40 | 14.06 |
| ICSI | Efficiency | 1.000 | 1.000 | 1.000 |
| | Effectiveness | 0.992 | 0.029 | 0.029 |
| | N | 4.06 | 36.91 | 6.00 |



## 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 effeciently



## 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!

Table 1. PD and FAR for various algorithms

| | | Probe | DoS | U2R | R2L |
|---|---|---|---|---|---|
| MLP | *PD* | 0.887 | 0.972 | 0.132 | 0.056 |
| | *FAR* | 0.004 | 0.003 | 5E-4 | 1E-4 |
| GAU | *PD* | 0.902 | 0.824 | 0.228 | 0.096 |
| | *FAR* | 0.113 | 0.009 | 0.005 | 0.001 |
| K-M | *PD* | 0.876 | 0.973 | 0.298 | 0.064 |
| | *FAR* | 0.026 | 0.004 | 0.004 | 0.001 |
| NEA | *PD* | 0.888 | 0.971 | 0.022 | 0.034 |
| | *FAR* | 0.005 | 0.003 | 6E-6 | 1E-4 |
| RBF | *PD* | 0.932 | 0.730 | 0.061 | 0.059 |
| | *FAR* | 0.188 | 0.002 | 4E-4 | 0.003 |
| LEA | *PD* | 0.838 | 0.972 | 0.066 | 0.001 |
| | *FAR* | 0.003 | 0.003 | 3E-4 | 3E-5 |
| HYP | *PD* | 0.848 | 0.972 | 0.083 | 0.010 |
| | *FAR* | 0.004 | 0.003 | 9E-5 | 5E-5 |
| ART | *PD* | 0.772 | 0.970 | 0.061 | 0.037 |
| | *FAR* | 0.002 | 0.003 | 1E-5 | 4E-5 |
| C4.5 | *PD* | 0.808 | 0.970 | 0.018 | 0.046 |
| | *FAR* | 0.007 | 0.003 | 2E-5 | 5E-5 |



## 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* occurences 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

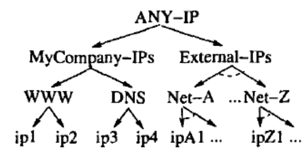
- Attainable degree of automation very low
  - <1% of alarms could be handled automatically based on previous episodes
- Tends to produce large number of irrelevant/redundant patterns
- Many patterns difficult to interpret

## 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




a) Generalization hierarchy for IP addresses.

| SrcIP | DstIP | Count |
|---|---|---|
| ip1 | ip4 | 1000 |
| ip1 | ipA1 | 1 |
| ip1 | ipB1 | 1 |
| ⋮ | ⋮ | ⋮ |
| ip1 | ipZ1 | 1 |
| ipA1 | ip4 | 1 |
| ipB1 | ip4 | 1 |
| ⋮ | ⋮ | ⋮ |
| ipZ1 | ip4 | 1 |

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***