

Learning from a flaw in a naive-Bayes masquerade detector

Kevin Killourhy and Roy Maxion

Background

- A **masquerader** gains access to another user's account for malicious purposes
- Detecting masqueraders is typically done with machine learning (comparing masquerader behavior to normal behavior)
- Naive-Bayes detectors are found to work well

Problem

- We found two **super-masqueraders**: never detected, no matter whose account was attacked
- Can any adversary become a super-masquerader? How do we stop them?
- How do we even approach this sort of problem?

Approach

1. **Error analysis**: Examine the data that caused the errors; then form hypotheses
2. **Controlled testing**: Investigate the hypotheses under controlled, synthetic conditions to establish cause
3. **Algorithmic analysis**: Explain causal effects
4. **Use domain knowledge**: Fortify the detector

Naive Bayes

- Historically, detectors focus on UNIX commands
- Naive Bayes scores a block of commands

$$\text{score}(c_1, \dots, c_b) = \frac{\sum_{i=1}^b \log \frac{s[c_i] + p}{s_t + \alpha p}}{\sum_{i=1}^b \log \frac{n_m[c_i] + p}{n_t + \alpha p}}$$

- b : # cmds in block p : pseudocount
- m : # nonself users α : alphabet size
- $s[c_i]$: # times i -th command is in self data
- $n_m[c_i]$: # times i -th command is in nonself data

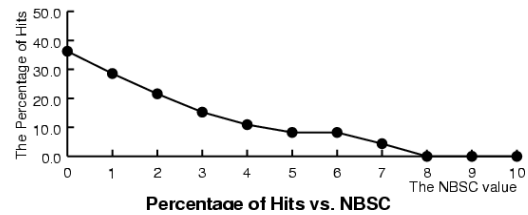
- This score compares the probability that the command came from the user's data (self) vs other data (nonself)
- A score higher than threshold sets off an alarm

1. Forming a hypothesis

- Both of the super-masqueraders used a peculiar (never-before-seen) invocation of `rlogin`
- Hypothesis:
Never-before-seen commands (NBSCs) can be used by an adversary to evade detection by naive Bayes

2. Controlled experimentation

- We generated 13,200 datasets to test how increasing the number of NBSCs affects naive Bayes under a range of conditions.
- Under all conditions, the hit rate goes to zero



3. Algorithmic analysis

$$\text{score}(c_1, \dots, c_b) = \frac{k \cdot \log(p) + b \cdot \log(s_t + \alpha p) + \sum_{i=1}^{n-k} \log(s[c_i] + p)}{k \cdot \log(p) + b \cdot \log(m s_t + \alpha p) + \sum_{i=1}^{n-k} \log(n_m[c_i] + p)}$$

• k : # of NBSCs in block

- NBSCs reduce the influence of other commands (blue) and bias toward self (red) dominates

4. Fortifying the algorithm

- An adversary could use symlinks or aliases to turn their attack into NBSCs
- This exploit can be prevented by adding an NBSC detector to naive Bayes

Naive Bayes	Misses	False Alarms	Cost
Simple	17.9%	5.7%	.236
Fortified	11.2%	6.3%	.175

- Average cost of using the fortified detector decreases by 34.7% over the simple detector
- More importantly, worst masquerader-miss-rate decreases by 66% (from 100% to 34%)