Anomaly Detection

Brian Palmer

What is an anomaly?

- the normal behavior of a process is characterized by a model
  - Deviations from the model are called anomalies.
- Example
  - Applications versus spyware
  - There is a model of what using the computer involves; if the system notices communication with strange hosts, it's an anomaly
  - Attacks on networks

Detecting Anomalies

- Anomalies are useful when nothing is known about intrusions
  - Modern systems like IDES combine anomaly detection with known intrusion databases
- Two ways of modelling normal behaviour
  - Models of normal behaviour can be the allowed patterns (positive detection)
  - Anomalous patterns (negative detection)
  - Which do you think would be better?

Positive and Negative Detection

- Early research focused on positive detection
  - It seems smaller and simpler
- Advantages of negative detection
  - More common in nature
  - Same amount of information
  - Easier to distribute

Algorithms/Anomaly Detection

- N-gram auditing
- Finite automata
- System call sense of self
- Artificial Immune System

N-Gram auditing

- Intrusions are correlated with abnormal behaviour
  - Events are logged into a symbolic audit trail
    - How to represent this? Simplify features
  - Then use an n- across the audit log to compare with positive training data

In a shell, for example, not “cat file1.c” but “cat <1>”

Finite Automata

- N-gram auditing is very simple
  - Processes are complicated
- Use finite automata instead
  - Hand constructing can be straightforward – or hard
  - Much better to let the computer do the work (e.g., Baum-Welch algorithm for HMM)
  - But HMM calculation is expensive

Training a Finite Automaton

- This approach outlined by Michael & Ghosh
- Works on a training set of nonintrusive patterns
  - It must accept every element in the training data
  - That's trivial – how can you do it easily? (Hint: one way is too weak, another is too strong)

What is a finite automaton

- (S,f)
  - A finite set *S* of states
  - A transition mapping *f* such that given a sequence of elements *I*, *f(s1,I)=s2* for some *s1,s2* in S
  - Generalization from the DFA/NFA that we're probably all familiar with

Using n-grams to construct it

- Each state is associated with one or more *n*-grams of audit information
  - *n* is a parameter of the algorithm
  - More than one *n*-gram for most states
- When we see a new *n*-gram, either create a new state, or reuse an existing state
  - Transition is the last *l* elements of the current *n*-gram
  - *l* is a parameter of the algorithm

Deciding to create a state

- Ask one simple question
  - "For the next *n*-gram, can the automaton already accommodate it?"
- The answer comes in three forms:

Creating a State: Form 1

- The current state has a transition matching last *l* elements of previous *n*-gram to a state associated with the new *n*-gram
    - Action: Done

Creating a State: Form 2

- The current state has a matching transition, but not to the correct state (or there is no matching state).
  - Create a state for the new *n*-gram if it does not exist
  - Create a transition from the current state to the new n-gram's associated state, using the last *l* elements of the previous *n*-gram

Creating a State: Form 3

- The current state has no outgoing edges that correspond to last *l* elements of previous *n*-gram.
  - If there is already a state assigned to the next *n*-gram, add a transition to it as previously
  - If not, we assign it to a compatible state
  - The authors quibble over good compatibility, but go with longest matching prefix
  - If there are no compatible states, then create a new state

Size of the automaton

- No *n*-gram has more than one state associated with it
  - Thus, no more than k^n states for a program with *k* unique audit events
  - Total number of edges is bounded by k^{n+l}
  - In practice, it is much smaller

Example

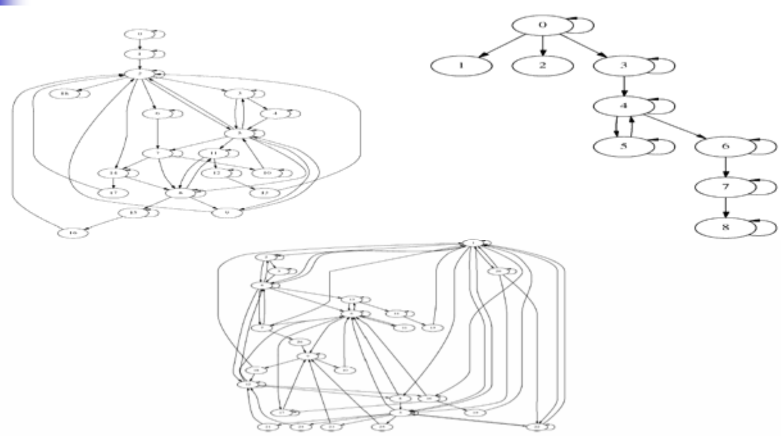


Fig. 1. Finite automata constructed for 1pr with $n = 2$, $\ell = 1$ on Week 2 of the Lincoln Labs data, constructed with $n = 7$, $\ell = 4$ on the same data, and constructed with $n = 2$, $\ell = 1$ for all seven weeks of data.

Confidence

- Rather than simply accepting or rejecting, we should have a confidence value in the automaton's assertion
  - If the current state exists, and there's a transition for *l*, then the confidence that it is an anomaly is 1-P(taking this transition)
  - P(taking this transition)=# of times it was taken in training/# of times the current state was encountered in training

Confidence 2

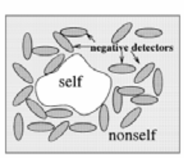
- More absolute values:
  - If the current state exists, but there's no transition, P(Anomaly)=1
  - If the current state is not defined (i.e., previous state had no correct transition), P(Anomaly)=0

Performance

- Note that this is used for real-time detection
  - Very efficient
  - Training is done in linear time

Immune System

- The human immune response is a driving metaphor
  - T-cells are grown in the thymus and accustomed to *self* peptides
  - Ones that react to the self peptides are censored; others are released into the body



Simple notion of self

- We want to tag known software runs with some identifier of self.
  - Any anomaly should interfere with these signatures
  - Normal runs of the program should not
    - But it should be able to run on arbitrary data

System calls

- System calls are easily tracked by the kernel for arbitrary programs
  - Already requires a context switch
  - Will be involved in any critical intrusion
- Claim: they form a useful "fingerprint" for intrusions

System call windows

- Similar to the n-gram auditing, we will use a *k* element window and slide it over a trace of system calls
  - But rather than pay attention to the exact order, we collect the k in the tail simply as valid successors to the *k* in the head.

Example: System Call Window

- Trace: open, read, mmap, mmap, open, getrlimit, mmap, close
- Let *k=3*

call	position 1	position 2	position 3
open	read, getrlimit	mmap	mmap, close
read	mmap	mmap	open
mmap	mmap, open, close	open, getrlimit	getrlimit, mmap
getrlimit	mmap	close	
close			

Mismatches

- The likelihood that it is an anomaly against a live run is found by counting the number of mismatches in sequences
- Maximum number of mismatches for a sequence of length *L* with lookahead of *k* is $k(L-k)+(k-1)+(k-2)+\ldots+1=k(L-(k+1)/2)$
- So # mismatches/Maximum # of mismatches = confidence of anomaly

Immune System as Algorithm

- Sequence of events form a string in a universe U
- We have a set RS of these strings; we can access only a sample S to train on
- Candidates are generated randomly and censored against S
  - We'll discuss the form of these candidates later on
- Those that fail to match any in S are retained as active detectors

Immune System Algorithm 2

- Each detector is independently generated
  - so it's probabilistic that, given sufficient of them, they'll detect anomalies
  - Works when given only positive examples to train against (why is that important?)
  - Done

Artificial Immune System

- Hofmeyr adapted this for an online, dynamic detector for network attacks
  - More like a real biological system
  - Immature, mature, and memory detectors present in system
  - Immature ones are deleted if they match a connection
  - Mature ones that are sufficiently discerning are promoted to memory detectors, with extended lifetime but lower threshold of activation

Artificial Immune System 2

- Activation thresholds work to prevent autoimmune disorders
  - Metaphor to proteins' avidity thresholds
  - Temporal clumping works for single host attacks
  - To handle distributed attacks, each successful detection lowers activation threshold
  - Each of these goes down over time

Form of Detector

- In this system, a detector is
  - An element of the set or
  - An r-chunk – length r string along a fixed position
- R-chunks are strictly more powerful than rcb matching

Different schemes

- Different choices are possible for most of the crucial parameters

Similarity of Sequences

- How close are two sequences of events?
- What would be a good way to classify their similarity?
- We want a distance metric

Sequence Similarity Metrics

- Hamming Distance
- Distance in n-space
  - Plot $x_1,\ldots,x_n$ and $y_1,\ldots,y_n$ as points in R^n
  - Requires same-size sequences, large dimensions, numerical approximation
  - Outliers grossly affect this
- Largest common subsequence
  - Define Sim(X,Y)=|LCS(X,Y)|/max(|X|,|Y|)
  - Also seems to be known as r-contiguous bits (rcb)
  - Various variants of this: linear filter, scaling, r-chunk

Summary

- I've gone over 4 basic approaches to detecting anomalies given positive training data.
  - These tend to be very efficient, but specialized, at detecting oddities in systems
  - Useful in many areas of (computer) security

References

- *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. PhD thesis, Purdue University, W. Lafayette, IN, August 2000.
- *A Sense of Self for Unix Processes*. Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, Thomas Longstaff.
- Bollobas, B., Das, G., Gunopulos, D., Mannila, H., `Time-Series Similarity Problems and Well-Separated Geometric Sets', Proc. of 13th Annual ACM Symposium on Computational Geometry, To appear, 1998. http://citeseer.ist.psu.edu/bollobas98timeseries.html
- "Immunity by Design: An Artificial Immune System" by Hofmeyr and Forrest