360° Unsupervised Anomaly-based Intrusion Detection

Stefano Zanero, Ph.D.
Post-doc Researcher, Politecnico di Milano
CTO & Founder, Secure Network S.r.l.

Black Hat Briefings – Washington DC, 01/03/2007
Presentation Outline

- Building a case for Anomaly Detection Systems
  - Bear with me if you already heard this rant :)  
  - Intrusion Detection Systems, not Software!
  - Why do we need Anomaly Detection?
- Network-based anomaly detection
  - Solving the curse of dimensionality
  - Clustering the payloads of IP packets
- Host-based anomaly detection
  - System call *sequence* analysis (done many times)
  - System call *argument* analysis (almost never)
  - Combining both, along with other ingredients
- Detecting 0-day attacks: hope or hype?
- Conclusions
A huge problem, since 331 b.C.

- The defender's problem
  - The defender needs to plan for everything... the attacker needs just to hit one weak point
  - Being overconfident is fatal: King Darius vs. Alexander Magnus, at Gaugamela (331 b.C.)

- Acting sensibly is the key ("Beyond fear", by Bruce Schneier: a must read!)

- "The only difference between systems that can fail and systems that cannot possibly fail is that, when the latter actually fail, they fail in a totally devastating and unforeseen manner that is usually also impossible to repair" (Murphy's law on complex systems)

- a.k.a. “plan for the worst !!!” (and hope)
Tamper evidence and Intrusion Detection

- An information system must be designed keeping in mind that it will be broken into.
  - We must design systems to withstand attacks, and fail gracefully (failure-tolerance)
  - We must design systems to be tamper evident (detection)
  - We must design systems to be capable of recovery (reaction)

- An IDS is a system which is capable of detecting intrusion attempts on the whole of an information system

- We need intrusion detection, despite what Gartner's so-called analysts think or say

- The question is: which type of IDS components do we need to answer our requirements?
**The big taxonomy: Anomaly vs. Misuse**

<table>
<thead>
<tr>
<th>Anomaly Detection Model</th>
<th>Misuse Detection Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Describes <strong>normal</strong> behaviour, and flags deviations</td>
<td>- Uses a knowledge base to recognize the <strong>attacks</strong></td>
</tr>
<tr>
<td>- <strong>Theoretically</strong> able to recognize any attack, also 0-days</td>
<td>- Can recognize only attacks for which a <strong>signature</strong> exists</td>
</tr>
<tr>
<td>- Strongly dependent on the <strong>model</strong>, the <strong>metrics</strong> and the <strong>thresholds</strong></td>
<td>- Problems for <strong>polymorphism</strong> (e.g. ADMmutate), as well as signature expressiveness and canonicalization issues</td>
</tr>
<tr>
<td>- Generates statistical alerts: “Something’s wrong”</td>
<td>- The alerts are precise: they recognize a specific attack, giving out many useful informations</td>
</tr>
<tr>
<td>- Difficult to use for automated reaction</td>
<td>- Can be easily used for automated reaction</td>
</tr>
<tr>
<td>- Has an ineliminable number of false positives</td>
<td>- Usually no false positives, but “noncontextual alerts” to be tuned out</td>
</tr>
<tr>
<td>- Evaded by “mimicry”</td>
<td>- Evaded by “strangeness”</td>
</tr>
</tbody>
</table>
Unsupervised learning

- At the Politecnico di Milano Performance Evaluation lab we are working on anomaly-based intrusion detection systems capable of *unsupervised learning*

- What is a learning algorithm?
  - It is an algorithm whose performances grow over time
  - It can extract information from training data

- Supervised algorithms learn on labeled training data
  - “This is a good event, this is not good”
  - Think of your favorite bayesian anti-spam filter
  - It is a form of generalized misuse detection

- Unsupervised algorithms learn on unlabeled data
  - They can “learn” the normal behavior of a system and detect variations (remembers something ... ?) *[outlier detection]*
  - They can group together “similar things” *[clustering]*
What is clustering?

- *Clustering is the grouping of pattern vectors into sets that maximize the intra-cluster similarity, while minimizing the inter-cluster similarity*

- What is a pattern vector (tuple)?
  - A set of measurements or attributes related to an event or object of interest:
  - E.g. a person's credit parameters, a pixel in a multi-spectral image, or a TCP/IP packet header fields

- What is similarity?
  - Two points are similar if they are "close"

- How is "distance" measured?
  - Euclidean
  - Manhattan
  - Matching Percentage
An example: K-Means clustering
Assign Instances to Clusters
Find the new centroids
Recalculate clusters on new centroids
Which Clustering Method to Use?

- There are a number of clustering algorithms, K-means is just one of the easiest to grasp.
- How do we choose the proper clustering algorithm for a task?
  - Do we have a preconceived notion of how many clusters there should be?
    - K-means works well only if we know K
    - Other algorithms are more robust
  - How strict do we want to be?
    - Can a sample be in multiple clusters?
    - Hard or soft boundaries between clusters
  - How well does the algorithm perform and scale up to a number of dimensions?
- The last question is important, because data miners work in an offline environment, but we need speed!
  - Actually, we need speed in classification, but we can afford a rather long training
Outlier detection

- What is an outlier?
  - It’s an observation that deviates so much from other observations as to arouse suspicions that it was generated from a different mechanism.

- If our observations are packets... attacks probably are outliers
  - If they are not, it’s the end of the game for unsupervised learning in intrusion detection.

- There is a number of algorithms for outlier detection.

- We will see that, indeed, many attacks are outliers.
Multivariate time series learning

- A time series is a sequence of observations on a variable made over some time
- A multivariate time series is a sequence of vectors of observations on multiple variables
- If a packet is a vector, then a packet flow is a multivariate time series
- What is an outlier in a time series?
  - Traditional definitions are based on wavelet transforms but are often not adequate
- Clustering time series might also be an approach
  - We can transform time series into a sequence of vectors by mapping them on a rolling window
A hard problem, then...

- A network packet carries an unstructured payload of data of varying dimension.
- Learning algorithms like structured data of fixed dimension since they are vectorized.
- A common solution approach was to *discard the packet contents*. Unsatisfying because many attacks are right there.
- We used **two** layers of algorithms, prepending a clustering algorithm to another learning algorithm.
- After much experimentation we found that a Self Organizing Map (with some speed tweaks) was the best overall choice.
The overall architecture of the IDS

First stage

Header
IP  TCP

Payload

Decoding

Clustering

Second Stage

Correlation
Recognising the protocols...
Recognising the attacks

- Let us look at HTTP (DPORT=80)
- Attack packets are in blue, normal packets in orange
- The characterization makes attacks outliers!
Outlier detection & results

- Using the Smart Sifter outlier detection algorithm
  - Detection Rate well above 70%
  - False Positive Rate around 0.03%
- Some thousands of false alerts per day
  - An order of magnitude better than other systems
  - Still, too much: we are working on it
- We will release the tool as a GPL Snort plug-in... I know, I've been promising for two years, but I'm just never satisfied...
ROC curve of our NIDS
HIDS: state of the art

- Host-based, anomaly based IDS have a long academic tradition, and there's a gazillion papers on them.
- Let us focus on one observed feature: the sequence of system calls executed by a process during its life.
- Assumption: this sequence can be characterized, and abnormal deviations of the process execution can be detected.
- Earlier studied focused on the sequence of calls:
  - Used markovian algorithms, wavelets, neural networks, finite state automata, N-grams, whatever, but just on the sequence of calls.
  - Markov models comprise other models.
- An interesting and different approach was introduced by Vigna et al. with “SyscallAnomaly/LibAnomaly”, but we'll see that in due time.
Time series learning (again)

- If a syscall is an observation, then a program is a time series of syscalls
- If our observations are descriptive of the behavior of systems... attacks probably are outliers
- Once again, definitions based on wavelet transforms are not adequate
- Markov chains give us an approach to model the SEQUENCE of system calls
  - Has been done a number of times
What is a Markov chain?

- A stochastic process is a finite-state, k-th order Markov chain if it has:
  - A finite number of states
  - The Markovian property (probability of next state depends only on \( k \) most recent states)
  - Stationary transition probabilities (not variable w/time)

- Probabilities, in a first-order chain with \( s \) states can be expressed as a square matrix of order \( s \)
  - In \( n \)-th order, with a order \( s^n \)

- They comprise other models
  - N-grams are simplified \( n \)-th order markov chains
  - FSA are simplified markov chains (almost ;)
  - Probabilistic grammars are Markov chains (probably)
An example of Markov chain

Markov Chain Models

transition probabilities

\[
\begin{align*}
\Pr(x_i = a \mid x_{i-1} = g) &= 0.16 \\
\Pr(x_i = c \mid x_{i-1} = g) &= 0.34 \\
\Pr(x_i = g \mid x_{i-1} = g) &= 0.38 \\
\Pr(x_i = t \mid x_{i-1} = g) &= 0.12
\end{align*}
\]
Training a Markov chain

- We can compute the likelihood of a sequence in a model with a simple conditional probability.
- We can build the model which fits a given sequence or set of sequences by calculating the maximum likelihood model, the one which gives the various observations the maximum probability.
- Can be done through simple calculations (problem of null probabilities), or through Bayesian ones.
- Comparison of probability of sequences of different length is difficult (can use the logarithm or other tricks to smooth).
Which Markov chain does this fit?

- Simple answer: you compute the likelihood
- If you need to compare multiple models, this is more complex
  - You need to take into account the prior probability, or probability of the model, since:
    \[ P(M|O) = \frac{P(O|M) \ P(M)}{P(O)} \]
  - \(P(O)\) is fixed and cancels out, but you usually don't know \(P(M)\): depending on the choice, you can have varying results
- S. Zanero, “Behavioral Intrusion Detection” explains the mathematical trick
SyscallAnomaly: analyzing the variables

- SysCall Anomaly, proposed by Vigna et al.
  - Each syscall separately evaluated on 4 separated models
    - (maximum) string length
    - Character distribution
    - Structural inference
    - Token search

- Each model is theoretically interesting, but exhibits flaws in real-world situations
  - Structural inference
    - Realized as a markov model with no probabilities...
    - Too sensitive!
  - Token search
    - No “search”, really: you must predefine what is a token
    - Again, no probabilities
Our proposal

- We evolved the models
  - Structural inference: abolished (halving false positives...)
  - Implemented a model for filesystem paths (depth – structural similarities, e.g. elements in common)
  - Token Search: probabilistic model
    - UID/GID specialization, considering three categories: superuser, system id, regular id

- Now, we wanted to add
  - Correlation among the arguments of a single syscall
    - Hierarchical clustering algorithm to create classes of use
  - Correlation among system calls over time
    - Through a proven, reliable Markov chain
Clustering system calls

- Clustering is the grouping of pattern vectors into sets that maximize the intra-cluster similarity, while minimizing the inter-cluster similarity.

- Here “pattern vectors” are the values of various models.

- We used a hierarchical agglomerative algorithm:
  - Pick up the two most similar items.
  - Group them.
  - Compute distance from the new group to other groups.
  - Repeat.

- What is similarity?
  - Two patterns are similar if they are “close”.
  - We had to define similarity for each model type.
    - e.g. is /usr/local/lib similar to /usr/lib? And to
Results of clustering

- The clustering process aggregates similar uses of a same system call
  - E.g.: let us take the open syscalls in fdformat:
    - `/usr/lib/libvolmgt.so.1, -rwxr-xr-x`
    - `/usr/lib/libintl.so.1, -rwxr-xr-x`
    - `/usr/lib/libc.so.1, -rwxr-xr-x`
    - `/usr/lib/libadm.so.1, -rwxr-xr-x`
    - `/usr/lib/libw.so.1, -rwxr-xr-x`
    - `/usr/lib/libdl.so.1, -rwxr-xr-x`
    - `/usr/lib/libelf.so.1, -rwxr-xr-x`
    - `/usr/platform/sun4u/lib/libc_psr.so.1, -rwxr-xr-x`
    - `/devices/pseudo/mm@0:zero, crw-rw-rw-
    - `/devices/pseudo/vol@0:volctl, crw-rw-rw-
    - `/usr/lib/locale/iso_8859_1/LC_CTYPE/ctype,-r-xr- xr-x`

- Each of the clusters is a separate type of syscall (e.g. “open 1”, “open 2”, “open 3”)
A matter of sequence

We can now build a Markov chain which uses as states the clusters of syscalls, as opposed to the syscalls per se.

We can train the model easily on normal program executions.

Not static analysis, we would include bugs.

At runtime we will have three “outlier indicators”:

- The likelihood of the sequence so far
- The likelihood of this syscall in this position
- The “similarity” of this syscall arguments to the best-matching cluster

1) indicates likely deviation of program course

2) and 3) punctual indicators of anomaly
ROC curve of our HIDS
Putting it all together!

- **What do we have so far?**
  - A system which flags anomalous packets with an "outlier factor"
  - A system which flags anomalous syscalls on a host with a (set of) outlier factor(s)

- **How can we correlate these alerts, maybe even along with others?**

- **A process of alert stream fusion**
  1) Aggregation of alerts referring to the same event
  2) Correlation of events likely to be related
  3) Scenario awareness and high-level analysis

- **We addressed 1) and 2) until now**
Aggregating alerts

- Putting together alerts with common features (attacker, target, service...) and “near” in time
- Near = fuzzy concept
- More robust. Models uncertainty and errors as well!
False positive reduction

- We compare FPR and DR reduction while incrementing aggregation and suppression of alerts.
- Belief correction preserves from suppression alerts with high support.
Using “causality” to study correlation

- Granger test for causality
  - If some_data is better explained with some_other_data in input than it is by itself, then other_data causes data
  - More formally, if an AR model on the output fits worse than an ARX model with the input, then the input “causes” the output
  - ... Nobel prize for Economy.

- Some early researchers proposed it for correlation, and we tried

- Results are (IMO) unconclusive, but the approach seems reasonable
A word of caution about “results”

- See my presentation at BH Fed on why the evaluation of intrusion detection systems is mostly useless as of now.
- Additionally, testing “correlation” would need us to know what we are looking for, but that's matter for another presentation in the future...
Conclusions & Future Work

- Conclusions:
  - IDS are going to be needed as a complementary defense paradigm (detection & reaction vs. prevention)
  - In order to detect unknown attacks, we need better anomaly detection systems
  - We can successfully use unsupervised learning for anomaly detection in an host based environment using
    - System call sequence
    - System call arguments
  - We can successfully aggregate alerts in an unsupervised fashion. Correlation needs more work!

- Future developments:
  - Correlation :)
  - Integrating the host based solution to become an IPS, maybe using CORE FORCE?
  - Real-world evaluation, perhaps in the framework of a proposed European FP7 project
Thank you!

Any question?

I would greatly appreciate your feedback!

Stefano Zanero
zanero@elet.polimi.it
www.elet.polimi.it/upload/zanero/eng