Anomaly Detection and Honeypots: relationships, anyone?

Stefano Zanero

Politecnico di Milano
Dipartimento di Elettronica e Informazione
zanero@elet.polimi.it

TERENA Networking Conference 2008
Bruges, Belgium
20/05/2008

Outline

- Motivations of the work
- Network Anomaly Detectors usefulness in a honeynet
- Host Anomaly Detectors usefulness in a honeynet
- Conclusions and research perspectives

Intrusion detection systems vs. Honeypots

- An IDS is a system which is capable of detecting security violations on an information system
  - Host-based vs. Network-based
  - Misuse based vs. Anomaly based
- An honeypot is a system which only has value if it is compromised.
  - Anomaly based effort (but everything which happens is anomalous)
  - In-depth review through forensics

Anomaly, Misuse and the Honeynets

- Traditionally, misuse based systems deployed as part of honeywalls for egress filtering and containment
  - Since misuse based detection tries to define "attacks", it cannot deal properly with:
    - New attacks (zero-days)
    - Polymorph attacks (can be exploited in many, even infinite, ways)
    - Insiders abusing their privileges ("non-attacks", but still security violations)
  - $\Longrightarrow$ it is almost useless in the context of honeynet research today
- Anomaly based systems learn by contrast with normality
  - On a honeynet, anything $\neq 0$ is anomalous
  - Anomaly detection seems useless as well
- So, that is the end of my talk...

Lessons from Anomaly Detection in the context of Honeynets

Some key features of anomaly detectors are actually useful also in this context:

Outlier detection: the ability of detecting "attacks" over "normal behavior" can be exploited to sift through a large volume of anomalous traffic and find egregious events

Generalization: the modeling techniques we use in anomaly detection to create models of normal behavior can be used to capture generic models of attacks

Behavioral Clustering: some techniques of behavioral clustering under development in anomaly detection can be used to create taxonomies of attacks or malware.

Existing example: the Leurrecom project

- Leurré.com
  - www.leurrecom.org
  - operated by Institut Eurécom
  - Broad network of honeypots covering more than 30 countries
  - Architecture of distributed low-interaction honeypots and a central server, using ScriptGen
  - All traces captured on each platform are uploaded on a daily basis into a centralized relational database
  - All project partners can access the whole database. Simple queries are open also to the outside
- On this dataset, Marc Dacier and his group discovered and published several clique-based algorithms that can automatically discover interesting attack patterns

Our current work: applying payload clustering

- In our unsupervised network anomaly detector, named ULISSE, we developed a way to use Kohonen's Self Organizing Maps to
  1. Preserve information about the "similarity" between packets
  2. Separate packets from different protocols in different groups
  3. **Separate packets with anomalous or malformed payload from normal packets**
- We demonstrated the SOM to be able to handle data at wire speed, and to be able to generalize well and divide different types of packets
- We are currently exploring if such characterization will help in quickly pre-analyzing network dumps from honeynets, e.g. grouping together similar traffic from scanners

System call anomaly detection, honeypot-style

- S^2A^2DE (Syscall Sequence and Arguments Anomaly Detection Engine)
  - A probabilistic, Markovian model of the behavior of processes
  - System call content analysis, with argument clustering
- We use the Markov model of a specific application to see if its instances behave correctly
- $\Longrightarrow$ We can use models generated from samples to recognize applications
- This also means that we can use this method, for instance, to classify malware based on the output of a sandboxed execution

A bayesian framework for the generation of Markov models

- In a previous research, we built a mechanism for generating an "adequate" number of Markov models directly from observations
- Let M be a generic model and O a sequence of observations, $P(M|O) \propto P(O|M)P(M)$
- If we have a set of I models $M_1, M_2 \ldots M_I$, the most likely model for the sequence of observations O is given by: $max_i\, P(M_i|O) = max_i\, P(O|M_i)\, P(M_i)$
- We need an appropriate *prior* $P(M_i)$ for this metamodel
- Common decomposition: $P(M_i) = P(\theta_i|M_s)P(M_s)$, with $P(\theta_i)$ probability of the parameter set given a structure and $P(M_s)$ probability of the structure. Too complex.

Non-informative prior criterion for model generation

- Non-informative prior criterion:

$$P(M_i) = \left(\frac{|O_i| + |O_k|}{(\sum |O_i|) + |O_k|} \right)^{log(|O_k|)}$$

  O_i: union of sequences that have generated model M_i
- $log(|O_k|)$ balances the fact that different length of observation strings generate different orders of magnitude in posterior probability
- Also a criterion for the iterative creation of new models. If M_{I+1} is a new model built on the new observations O_k, we would choose: $max_i P(M_i|O_k) = max_i P(O|M_i)P(M_i)$, defining:
  $P(M_{I+1}) = \left(\frac{|O_k|}{(\sum |O_i|)+|O_k|} \right)^{log(|O_k|)}$
- Meaning: bias towards more general models instead of more fitting (higher posterior probability $P(M_i|O_k)$) but less general ones.
- But this does not give us control on "how much" we want to generalize or not

Bayesian Model Merging

- A *merging* step, a la Rabiner, tries to find couples of models M_i, M_j s.t., denoting with $M_{i,j}$ the "merged" model and with O_i and O_j the observations associated with M_i and M_J:

$$P(O_i \cup O_j | M_{i,j}) P(M_{i,j}) > P(O_i | M_i) P(M_i)$$
$$P(O_i \cup O_j | M_{i,j}) P(M_{i,j}) > P(O_j | M_j) P(M_j)$$

- Selecting models to merge: use distance $D(M_i, M_j) = 1/T[logP(O^{(i)}|M_i) - logP(O^{(i)}|M_j)]$, where $O^{(i)}$ is a sequence of observations generated by model M_i. We are trying to find heuristic definitions that are not so heavy and yield good results

System call anomaly detection for Forensic uses

- In a recent work we demonstrated how S^2A^2DE can be exploited to detect in-memory injection of attack code and process substitution
  - Helps counter anti-forensic techniques
  - Creates an audit trail
- In many honeynet setups, the honeynet has no background activity so the usual approach is to log and analyze everything; however, this can be impractical
- In this case, an anomaly detector such as S^2A^2DE can be used to create heightened forensic trails of interesting activities; or to sift through historical forensic trails and spot interesting events by a recursive training of the detector to recognize already analyzed sets of activities
- "Artificial Ignorance" (M. Ranum) on steroids

Conclusions

- We have briefly shown how lessons drawn from anomaly-based IDS can be of use in honeypot research activities
  - finding interesting events in a larger volume of logs
  - creating models of attack behavior
  - creating taxonomies of attacks or malware through behavior analysis
- This quick overview is meant to raise questions, more than answers, and is dense with future, or ongoing work.
- Forward-looking statements standard warning applies :-)

Thanks for your attention !

- Any questions?
- You can reach me at `stefano.zanero@polimi.it`
- ULISSE materials can be downloaded at `http://trac.elet.polimi.it/ulisse`
- S^2A^2DE materials can be downloaded at `http://trac.elet.polimi.it/ssaade`