Design, Implementation, and Validation of a Self-Learning Intrusion Detection System

Davide Adami, Christian CALLEGARI, Stefano Giordano, Giada Landi and Michele Pagano

Department of Information Engineering
University of Pisa

Monitoring, Attack Detection and Mitigation
28-29, September 2006
Tubingen, Germany
Outline

1. SLIDS Architecture
   - TCP Modules
     - TCP/Markov Module
     - TCP/SLR Module
   - ICMP Module
   - UDP Module
   - Correlators and Classifiers

2. Experimental Results
SLIDS Architecture

- Software tool
- Network IDS
- Anomaly based IDS
- Centralized architecture
- State-full and Stateless approach
- Modular implementation, to allow extensibility and customizability
Input: libpcap files

Drop:
- Non IP packets
- Packets whose protocol field value differs from 1 (ICMP), 6 (TCP), and 17 (UDP)

Forward packets to the correct module
TCP/Markov Module

TCPDUMP → Traffic Filter → ICMP/SLR → UDP → TCP/Markov → TCP/SLR → Security Event Correlator → Security Event → TCP Classifier → Alarm

Security Event → Security Event Correlator → Security Event → UDP Classifier → Alarm
TCP/Markov Module

- Idea: Model TCP connections by means of Markov chains
- State-full approach
- The IP destination address and the destination port number are used to identify a connection
- State space is defined by the possible values of the TCP flags (6 bits)
- The value of the flags is used to identify the chain transitions
- A value $S_p$ is associated to each packet according to the rule

$$S_p = syn + 2 \cdot ack + 4 \cdot psh + 8 \cdot rst + 16 \cdot urg + 32 \cdot fin$$
TCP/Markov Module - Training phase

Calculate the transition probabilities

\[ a_{ij} = P[q_{t+1} = j | q_t = i] = \frac{P[q_t = i, q_{t+1} = j]}{P[q_t = i]} \]
TCP/Markov Module - Detection phase

- Given the observation \((S_1, S_2, \cdots, S_T)\)
- The system has to decide between two hypotheses
  
  \[ H_0 : \text{normal behaviour} \]
  
  \[ H_1 : \text{anomaly} \]

- It calculates the logarithm of the Likelihood Function
  
  \[
  \logLF(t) = \sum_{t=R+1}^{T+R} \log(a_{S_tS_{t+1}})
  \]

- It calculates its temporal “derivative”
  
  \[
  D_w(t) = \left| \logLF(t) - \frac{1}{W} \sum_{i=1}^{W} \logLF(t - i) \right|
  \]
TCP/Markov Module - Detection phase
TCP/Self Learning Rules Module

TCPdump → Traffic Filter → TCP/Markov → TCP/SLR → Security Event Correlator → TCP Classifier → Alarm

TCPdump → Traffic Filter → ICMP/SLR → Security Event Correlator → ICMP Classifier → Alarm

TCPdump → Traffic Filter → UDP → Security Event Correlator → UDP Classifier → Alarm

SLIDS Architecture
Experimental Results

TCP Modules
ICMP Module
UDP Module
Correlators and Classifiers
TCP/Self Learning Rules Module

- **Idea:** Self construct rules, observing training data
- Stateless approach
- It only analyzes the following packets:
  - SYN packet
  - First 100 packets of each connection
  - FIN packet
- For each of the selected packets, it analyzes the first 64 bytes (commonly 20B IP + 20B TCP + 24B payload)
- These bytes, considered 2 by 2, are called attribute $A_i$
- It constructs rules of the kind:
  
  
  $$\text{if } A_1 = v_1, A_2 = v_2, \ldots, A_k = v_k$$

  
  $$\text{then } A_{k+1} \in V = \{ v_{k+1}, v_{k+2}, \ldots, v_{k+r} \}$$
TCP/Self Learning Rules Module - Training Phase

**Rule Generation**

- Randomly select a subset $S$ of 100 packets
- Randomly select packets 2 by 2 from $S$, up to obtain 1000 pairs
- For each pair compute up to 4 rules, that are satisfied by both the packets
- Up to 4000 rules are generated
TCP/Self Learning Rules Module - Training Phase

Rule Testing

- Discard the redundant rules
- Discard rules with a low value of $n/r$, where $n$ is the number of packets that satisfy the antecedent and $r$ is the number of values the consequent can assume
- Test the generated rules on the whole dataset
- Discard rules whose value of $r$ increases during the last 10% of the training period:
  - Reduce potential false alarms
  - Last 10% of the training data is attack free
  - If given to the system as detection traffic (training performed on the 90% of the training data), each increase in $r$ would generate an alarm
TCP/Self Learning Rules Module - Detection Phase

- The first 64B of each packet are analyzed
- If a rule is not matched, a security event is generated
- For each broken rule the anomaly score is $t \times \frac{n}{r}$, where $t$ is the number of packets since the rule was last violated
- Final anomaly score is the sum of those computed for each broken rule
Analogous to TCP/SLR module

The approach is independent of the analyzed protocol

Attributes are processed, without considering neither their position in the packet nor their content
UDP Module

- Only perform some checks on the well-known port numbers
- Reveal an intrusion if a packet is directed to a port never seen during the training phase
- ...work in progress: apply a method similar to the SLR module
Correlators

- Reduce false alarm number
- Only consider one security event for each hypothetical attack
- All other security events related to that attack inside $x$ seconds are dropped
- ...work in progress: study correlation between the security events
Classifiers

- Receive the security events and decide if an alarm has to be generated
- Decision is taken on the basis of a threshold classifier
- Thresholds are set by means of MonteCarlo simulation
- ...work in progress: neural classifiers
Experimental Results

DARPA 1999 Data Set

Laboratory Network Traffic
Thank You!

Any Question?