

# **Comparing an Immunological and a Rule-Based Intrusion Detection Method**

John Hall

## **Abstract**

This paper compares an immunological based detection style such as the one used by ISNIDS to a simple rule-based detection style. Both Detection components were built into a common framework. We discuss our designs in considerable detail. Both systems were fed normal and attack data. Our results show that both methods have merit. We conclude with a discussion of the theoretical effectiveness of both styles.

## **Introduction**

Our goal of this project was to compare misuse and anomaly styles of intrusion detection. We have had some experience with Immunology based systems in our previous system ISNIDS, so we choose to use an Immunology based detection method for anomaly based detection. We choose to compare this style with a rule-based misuse style. We feel these are a good match because both methods are stateless and therefore can use similar detectors.

In order to perform a reasonable assessment of these methods, we needed to consider a number of actual attacks. We choose to handle password guessing, masquerading, scanning, and illegal data access attacks. We designed both detection systems with these attacks in mind.

## **Experiment**

We tested both detection modules under both normal and attack conditions. The normal condition data was generated over a two and a half week period. Over this period all audit data was saved into files. These files were later sent to both detection components. We also generated two attacks based on each threat considered. The audit data collected over the duration of these attacks was also saved and later fed into the detection components. Our attacks were, a nearby port scan, a remote port scan, password guessing through a tunnel from outside the network, password guessing from inside the network, masquerading through a tunnel from outside the network, masquerading from inside the network, external attempts to locate private data, and internal attempts to locate private data. The audit data from each attack was sent to each system independently.

## **Architecture and Implementation**

The architecture used for these experiments is shown in figure 1. We designed our system to read raw audit events either from a log file or live events. The sensors component merged the audit events from each individual sensor. The events were then grouped into sessions by the packager component. This grouping was done based on similar times, sources, and users for the events. The contents of a session are shown in figure 2. For the purposes of evaluating the detection component, these sessions were also saved with a time stamp. These sessions were passed to the detection component. Also for the purposes of evaluating the detection component, the output of the detection component was saved with a time stamp. No other IDS activity was done for this experiment.

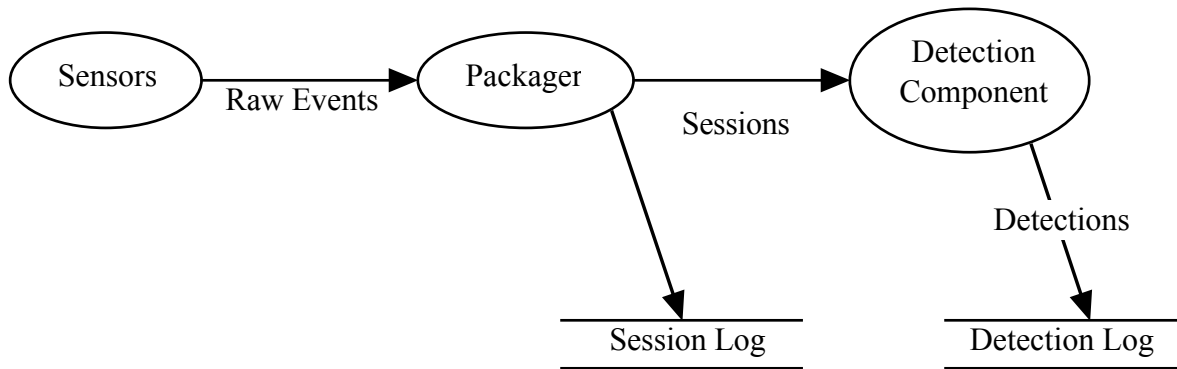


Figure 1. Experimental Architecture

USER - Set of Strings  
 SOURCE- Set of IP Addresses  
 DESTINATION- Set of IP Addresses  
 NWPROTOCOL- Set of Packet Types  
 PROTOCOL- Set of Protocol Port Numbers  
 TTL- Set of Packet Time-to-Live Values  
 LOGONS- Number of Successful Logons  
 LOGOFFS- Number of Logoffs  
 AUTH FAILURE- Number of Logon Failures  
 HTTP RESULT- Set of HTTP Result Codes  
 HTTP PROTOCOL ERRORS- Number of Protocol Errors  
 HTTP FILE ERRORS- Number of "File Not Found" Errors  
 HTTP ACCESS ERRORS- Number of "Forbidden" Errors

Figure 2. Session Contents

The immunology-based detector was trained off-line using the previous month's training data. The details of this training process are outside the scope of this document. See the initial release of ISNIDS for more information. Like the original release of ISNIDS, this detector used 2048 individual detectors. Each detector could match a specific value in each of the attributes listed in figure 2. Additionally, any attribute could be ignored. Each detector will be compared to each session as it is received. Each field of the detector will be tried in turn until the detector fails to match the session. The session will be considered an attack if it is matched by at least one detector.

The rule-based detector used a simple language to encode the rules. The actual rules file used for these tests can be found in appendix A. The language used to describe rules offers considerably more flexibility. The language used to encode these rules is described by the lex and yacc files found in the "/secondary/source/rulebased/" directory of the project. Each rule will be compared to each session as it is received. The session will be considered an attack if it is matched by at least one rule.

## Assessment

As mentioned, the normal data was collected from the environment over a two and a half week period. This data was intended to test for false positives. Each attack reported in this data was analyzed manually.

The immune detector detected 400 attacks in this data. The first 10 of these attacks are shown in appendix B. The rest are available in “/test/detection/results/immune\_new\_logs/Detections.txt”. At over 20 detections per day, this seems unwieldy. However, by our original definition of an attack, almost all of these are doorknob rattling. A vast majority of these are UDP requests to port 137. I believe these are NetBios commands initiated by Nimbda. I also found 8 TCP requests to Microsoft SQL Server port 1433 and 1434; 4 HTTP Protocol Error alerts that I believe were initiated by Code Red; 2 Telnet requests; 2 HTTPS requests; as well as a SOCKS (1080), SMTP (25), Location Service (135), ICMP ping, NetBios Session Service (139), and an unknown TCP port 33198 request. The immunology based detector found these all to not be normal traffic. These fall into my definition of an attack, but there are too many to deal with. I believe adding a rule based filter on the detections would make sense. There was one detection of interest. Detection number 7 (highlighted in appendix B) was packaged incorrectly. The packager merged legitimate activity with a doorknob rattling attack. The detector correctly identified the doorknob rattling portion as an attack. However, this may be a danger if an automated response may affect the legitimate activity. To summarize, I believe there was only one false positive made by the immunology-based detection engine when run on these 17 days of data.

The rule-based detector detected only 2 attacks in this data. Those detections are listed in appendix C. The first appears to be an attack. It is interesting that the immune-based detector did not detect this attack. The training data probably contained a similar attack. The second detection is the same incorrectly packaged event that the immune-based detector found. Both systems flagged this session as an attack. To summarize, the rule-based detector only made one false positive when run on these 17 days of data. The rule-based system did not detect the doorknob-rattling attacks simply because I did not write a rule with these attacks in mind. This also seems acceptable behavior, because I consider these doorknob-rattling attacks normal, and am not particularly interested in most of them.

Both the immune-based and rule-based detectors performed exceptionally well in this test when evaluated with respect to number of false positives. They both incorrectly identified the same session as an attack.

The next step of the test plan was to feed attack data to both detection systems. The 8 attacks mentioned above were used. Figure 3 shows the detection times for each attack. These times were measured from the time the session was first passed from the packager to the detection component. In each case, an additional 20 seconds were used from the time the sensors generated data until the packager provided a session. Almost all of the detections occurred in less than a second. This seems like more than adequate for a low-use network. I believe the packaging time is

the main candidate for improvement. The only exception was the immune-based system with attack 1. The detector did not detect an attack based on either of the first three sessions. Since the packager waits 20 more seconds between each update of the session, the total time was significantly higher. Both detection schemes detected six attacks and missed two attacks. Based on these tests, this gives each detector a 25% false negative rate. This indicates that each detection scheme offers fairly good protection. However, the false negative rate is still a little higher than I am comfortable with. It is also interesting the the attacks missed by the rule-based system were the masquarding attacks.

	Attack	Immune-Based	Rule-Based
1	Nearby Portscan	41s	0s
2	Remote Portscan	0s	0s
3	Password Guessing Through Tunnel	0s	0s
4	Password Guessing From Inside	Not Detected	0s
5	Masquarding Through Tunnel	Not Detected	Not Detected
6	Masquarding From Inside	0s	Not Detected
7	External Attempts to Locate Private Data	0s	0s
8	Internal Attempts to Locate Private Data	0s	0s

Figure 3. Detection Times

Figure 4 shows the total processing time used by each detection style for each attack. This time includes the time used by the sensors and the packager. This data shows that the immune-based system takes from 0.1 to 0.3 seconds longer to perform detection on one session. When compared with the time currently used by the sensors and the packager, this difference does not seem particularly significant. The reason for the difference is simple. The immune-based system is checking over 2000 detectors for a match, while the rule-based system is checking only 8 rules for a match. Both systems could be optimized. However, it seems that the other components should be optimized first.

	Attack	Immune-Based	Rule-Based
1	Nearby Portscan	80.63s	80.43s
2	Remote Portscan	81.51s	81.20s
3	Password Guessing Through Tunnel	0.51s	0.34s
4	Password Guessing From Inside	0.67s	0.55s
5	Masquarding Through Tunnel	0.35s	0.28s
6	Masquarding From Inside	0.90s	0.77s
7	External Attempts to Locate Private Data	0.61s	0.51s
8	Internal Attempts to Locate Private Data	0.74s	0.68s

Figure 4. Total CPU Usage

## **Conclusion**

Both rule-based and immune-based detection schemes offer some benefits. A rule-based system is easy to configure as to what is and what is not an attack. An immune-based detection system is better at detecting the unexpected attacks. The immune system is often better at noticing some patterns. However, the immune-based system may miss some obvious attacks and alert on some allowed activities. My tests found that the two systems performed comparably well. Each had some weaknesses and some strengths. I would recommend combining both detection methods to maximize the effectiveness of an IDS.

## **Appendix A**

### **Misuse Detection Rules**

```
// Simple check on how many logon failures occurred.
IF not (#LOGON >=3) and (#AUTH_FAILURE >= 4) THEN ALARM
IF #AUTH_FAILURE >= 7 THEN ALARM
// Simple check on HTTP errors.
IF #HTTP_PROTO_ERROR >= 4 THEN ALARM
IF #HTTP_FILE_ERROR >= 4 THEN ALARM
IF #HTTP_ACCESS_ERROR >= 2 THEN ALARM
// If someone is doing things on many different ports,
// they are probably scanning.
IF #PROTO >= 20 THEN ALARM
// The only logon names I know about are "root" and "jhall".
IF not (      USER IS EMPTY or
          USER CONTAINS "root" or
          USER CONTAINS "jhall" ) THEN ALARM
// I only logon from home and from school.
IF not USER IS EMPTY and not ( SOURCE IS EMPTY or
                               SOURCE contains 127.0.0.1:32 or
                               SOURCE contains 10.0.0.0:24 or
                               SOURCE contains 192.0.0.0:8 ) THEN ALARM
```

**Appendix B.**  
**First 10 (of 400) Detections by Immune Detector.**  
**(Recieved From 2.5 Weeks of Data.)**

Event 1 (Wed Dec 18 14:10:08 2002): SESSION Source: 216.227.86.213 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 109 DETECTOR Source: 216.239.80.0:21  
NwProtocols: UDP TTL: 96-111

Event 2 (Wed Dec 18 14:10:09 2002): SESSION Source: 213.7.50.28 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 111 DETECTOR Source: 213.6.0.0:15 NwProtocols:  
UDP TTL: 111

Event 3 (Wed Dec 18 14:10:09 2002): SESSION Source: 61.217.3.143 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 114 DETECTOR Source: 61.216.0.0:15 NwProtocols:  
UDP TTL: 114

Event 4 (Wed Dec 18 14:10:09 2002): SESSION Source: 24.232.237.55 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 114 DETECTOR Source: 24.232.232.0:21  
NwProtocols: UDP TTL: 114

Event 5 (Wed Dec 18 14:10:09 2002): SESSION Source: 192.10.81.124 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 108 DETECTOR Source: 192.10.81.0:24  
NwProtocols: UDP TTL: 108

Event 6 (Wed Dec 18 14:10:09 2002): SESSION Source: 203.253.66.145 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 107 DETECTOR Source: 203.253.64.0:18  
NwProtocols: UDP TTL: 107

**Event 7 (Wed Dec 18 14:10:12 2002): SESSION User: jhall Source: 67.68.145.113 Dest:**  
**63.161.30.137 NwProtocols: UDP Protocol: 137 TTL: 113 Logoffs: 2 DETECTOR**  
**NwProtocols: UDP TTL: 112-127 Logon Count: 0-3 Logoff Count: 2-3**

Event 8 (Wed Dec 18 14:10:17 2002): SESSION Source: 66.191.241.19 Dest: 63.161.30.137  
NwProtocols: TCP Protocol: 1433 TTL: 112 DETECTOR Source: 66.191.240.0:22  
NwProtocols: TCP

Event 9 (Wed Dec 18 14:10:17 2002): SESSION Source: 61.176.162.223 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 112 DETECTOR Source: 61.177.48.0:20  
NwProtocols: UDP TTL: 112

Event 10 (Wed Dec 18 14:10:17 2002): SESSION Source: 61.189.130.187 Dest: 63.161.30.137  
NwProtocols: UDP Protocol: 137 TTL: 109 DETECTOR Source: 61.188.0.0:14 NwProtocols:  
UDP

**Appendix C**  
**Detections by Rule-Based Detector.**  
**(Recieved From 2.5 Weeks of Data.)**

Event 1: (Wed Dec 18 14:29:18 2002) SESSION Source: 211.161.25.17 Http Access Errors: 6  
RULE(5) IF #HTTP\_ACCESS\_ERROR >= 2 THEN ALERT

**Event 2: (Wed Dec 18 14:30:22 2002) SESSION User: jhall Source: 67.68.145.113 Dest:  
63.161.30.137 NwProtocols: UDP Protocol: 137 TTL: 113 Logoffs: 2 RULE(8) IF (not (USER  
IS EMPTY)) and (not (((SOURCE IS EMPTY) or (SOURCE CONTAINS 127.0.0.1:32)) or  
(SOURCE CONTAINS 10.0.0.0:24)) or (SOURCE CONTAINS 192.0.0.0:8))) THEN  
ALERT**