Table of Content

Abstract:

Firewalls and IDS (Intrusion Detection Systems) use packet classification for filtering incoming and outgoing packets. Traffic isolation through packet classification is important for devices like routers and firewalls that provide services like admission control, per-flow queuing and quality of service. These devices use a combination of algorithms and rules to classify packets and take appropriate action(s) which are defined based on access control lists. Most devices work well with smaller access control lists and lower data rates, but with most organizations demanding remote services and performance, this leads to complex networks with huge access lists and higher bandwidth. At the time of writing this report devices are able to push data rates of 10 Gig/sec. Higher data rates combined with large access control lists shrink the time devices have to inspect packets, hence leading to the problem of unclassified packets either being denied access or permit to enter organizational networks. Either case is not desirable as first leads to performance degradation and later leaves internal networks vulnerable to potential attacks.

Significant research and mathematical models are being developed from linear, hierarchical to geometric algorithms to find the fastest search algorithms and same can be said about processors. One aspect that hasn't seen significant research is rule consolidation. If two or more rules can be combined into one rule and still maintain the same security posture, which can reduce processing time significantly.

Introduction:

For a firewall or IDS system to be effective, administrators need to devise rules to maintain a selected security posture. Each incoming packet needs to be checked against the rule set. It is not an option to simply pass some packets untested because this compromises the selected security posture and could pose a significant security risk. Nor is it an option simply to drop traffic which the firewall does not have sufficient time to test. Blindly dropping packets being transported by a connection-oriented protocol like TCP will result in the source resending the packet until it times out or the packet is delivered. This could generate significant traffic and add to the increased congestion on the network.

As the complexity of network traffic increases the number of rules which must be checked for each packet increases. At the same time, firewalls and IDS systems are faced with increasing volumes of traffic so the overall task of checking every packet becomes even more difficult. Significant advances are being made in processor throughput as well as improving the underlying algorithms used in packet classification. Another avenue that needs to be explored is the simplification and regularization of the rulesets used in classification. We would like to see whether we can minimize the number of rules which have to be processed while maintaining a specified security posture.

Solution:

Develop a tool that will help administrators assess their security posture by determining the effectiveness of rulesets before they are employed in production environments. RuleAnalyzer will identify packets that match and don't match rules, in addition for the rules that match, it will also identify additional rules that packets would match. This

information can be especially helpful to an administrator in consolidating multiple rules into one rule and still maintain the required security posture.

It is just as important to know which packets don't match any rules. It's always challenging to deal with these packets because having blanket approach of either denying or permitting can have adverse effects. If unidentified packets make into the environment, it can compromise the security posture on the other hand if it's blocked a protocol like TCP will keep re-transmitting till it times out or the packet is delivered. This approach causes unnecessary traffic and adds to the congestion problem on the network.

Another issue the tool tries to address is the lack of agreed standards amongst network equipment manufacturers when it comes to creating access-control lists and capturing packets. In order for Rule Analyzer to be effective it needs to have the ability to understand rules created for other manufacturers and assess them. Although, not exhaustive, an attempt is being made to address problems arising from lack of standards. Considering the time limitations only two major manufacturers Cisco and Snort are being considered for porting rule sets and Wireshark and Windump are being considered for packet capturing tools. Rule Analyzer also makes a strong case for using XML (which is industry agreed standard for cross platform communication) as a standard for creating rulesets. It not only provides one with the flexibility of having custom naming standards, but also provides the ability to custom order rules as needed.

Tool Design and Description:

As described in the above section, Rule Analyzer's main purpose is to evaluate rules against a given traffic capture and provide insight by identifying rules that can be consolidated into one rule, thus leading to higher performance and security. Rule Analyzer is a Windows based application, developed in C Sharp and uses Windows .Net 2.0 framework as a foundation. The tool provides a number of capabilities namely, a packet capturing facility (limited to IP packets only), ability to load custom rulesets created using the proposed standard and also rules created using Cisco and Snort syntax. Rule Analyzer accepts captures by tools such as WireShark, Windump (with options a/e).

Rule Analyzer is broken in two sections, the first section deals with the capturing and analysis capability (figure1), where the user is able to select between having Rule Analyzer capture packets and analyze or load an existing capture from Rule Analyzer or another tool such as WireShark or WinDump for analysis. When selecting WireShark as an option, Rule Analyzer accepts .csv file and with WinDump it accepts outputs with option of (a/e) while capturing. Please note capturing packets isn't the main focus of the application, it has been included just as a quick and dirty capturing facility where an industry recognized tool such as WireShark or WinDump aren't available. The second section (figure2) addresses rules. When Rule Analyzer is invoked it loads a pre-designed ruleset. Users can view the ruleset and make changes and reload as per requirements. It gives users options to load rules in native RuleAnalyzer format, Cisco or Snort format. Please note currently RuleAnalyzer can handle only Cisco Standard and Extended acls.

The RuleAnalyzer native acls are an XML based format (figure3). There were two main reasons why XML was the format of choice. Firstly it gives the end user a great deal of flexibility in naming scheme and order for attributes of each element. Each acl entry is an Element and each element has attributes such as protocol, source ip, source port etc. For example if a user decides to call the protocol attribute as “proto”, rather than making changes to his/her rule files, he/she can configure RuleAnlayzer so that it will recognize “proto” as a valid attribute. In addition RuleAnalyzer is indifferent to ordering of attributes and lets end user define the ordering scheme as they see fit. This flexibility makes Rule Analyzer syntax independent, which will help users to confirm it to their liking and also users who are new to Rule Analyzer can start using it without a huge learning curve. Secondly XML format is an industry recognized standard and is portable from one vendor to another.

During the loading phase of rule analyzer, performs checks for hardware namely a network card, and loads a predefined ruleset into memory. Rule Analyzer loads rulesets into memory for faster execution and reduce the number of reads it needs to do during analysis. The predefined ruleset is a default ruleset that has catch all rules for protocols. After the load process is complete users have the ability to load custom rules. Once rules are loaded users can view rules loaded into memory and ensure that Rule Analyzer interprets the rules as they intended them to be. Caution must be taken during

the load process. Users must ensure that they check the right option when loading rules. For example if they intended to load a Cisco rulesets, users should ensure that they check the Cisco Rules radio button. Failure to do so will result in unexpected errors. As mentioned before Rule Analyzer currently handles standard and extended rules for Cisco. Rule Analyzer provides a user with three views, one is data grid view, and another is a text view and also an error log. The first two views are for users to confirm what they intended to load into the analyzer. The last view is to show the user the rules it wasn't able load, whether that be because the limitations within the analyzer or syntax error within the rule. This is for the user to investigate the problem and take corrective action. When loading XML rulesets users have the ability to configure analyzer to use attributes that they desire to use. Users need to configure the attributes before they load the rules.

Once the rules are loaded the user is ready to load the captured file and start the analysis. Care should be taken when loading captured files. Appropriate selection buttons need to be selected when loading the file, otherwise unexpected results will occur.

During the analysis phase Rule Analyzer creates a results file with a unique identifier that is a combination of date and time stamp. This is done to preserve results from previous runs. Once the analyzer has completed the analysis, it will summarize the results by protocol, by each of the rules (figure4). It will also write a detailed log of each packet and the corresponding rules that it matched (figure5). In addition it also provides the ability to search the analyzed file as desired. It gives the user to look for a certain combination of rules or list packets that matched certain number of rules. For example, user has the ability to pick from the drop down menu, 2, 3 … rules (figure7). Analyzer will list all the packets that matched that criteria. Users can drill down further can look a specific combination of rules that the packets match (figure9). As mentioned above Rule Analyzer accepts packet captured by WireShark and WinDump. During the analysis phase if Rule Analyzer finds packets that do not conform to the format it is expecting, it will write those packets to an error log (figure6). This functionality has been added to help future enhancements so the exceptional packets can be handled. After the analysis has finished it will categorize information by summarizing

Sample Results

Results from the analysis can be validated by having some priori knowledge of captured packets. Figure10 shows the analysis done by Wireshark that breaks captured packets by protocol. In this sample example the total number of packets is 10151, out of which 9729 are TCP, 388 packets are UDP and 34 are ICMP. After the analysis is done of the same capture (figure4), it does list as the total count to 10151, however it only list 9568 as being the TCP total and 356 as being UDP and counting 34 packets as ICMP and in addition the packets that it wasn't able to understand were 193. If we add all the individual protocols and the invalid packets (as per Rule Analyzer) it adds up to 10151.

Rule Analyzer also breaks the analysis by each rule and the number of packets matching those rules (figure11). You will note that the last rule 1-23 is a catch all rule for TCP and as expected every TCP packet in the capture matches that rule and same is true for UDP and ICMP packets.

Rule Analyzer also provides a search facility by which a user can search for all packets that matched a specific number of rules. Figure 6 shows all packets that matched two rules. User can also search for a specific combination of rules; this can be helpful in identifying overlaps in rules. With this information in hand, users can be sure there rules are unique and effective.

Conclusion

Rule Analyzer can be a powerful tool for administrators and researchers in evaluating rulesets. It can not only identify packets that don't match rules but also the ones that match multiple rules. This information can very valuable in enhancing the performance and securing the networks from unwanted packets. The XML format of the analyzer makes it very flexible to use and syntax independent. It's up to the user to decide the syntax and the order they like to view attributes. With very little learning users can perform useful analysis that can help them secure and enhance performance on their networks.

Appendix A

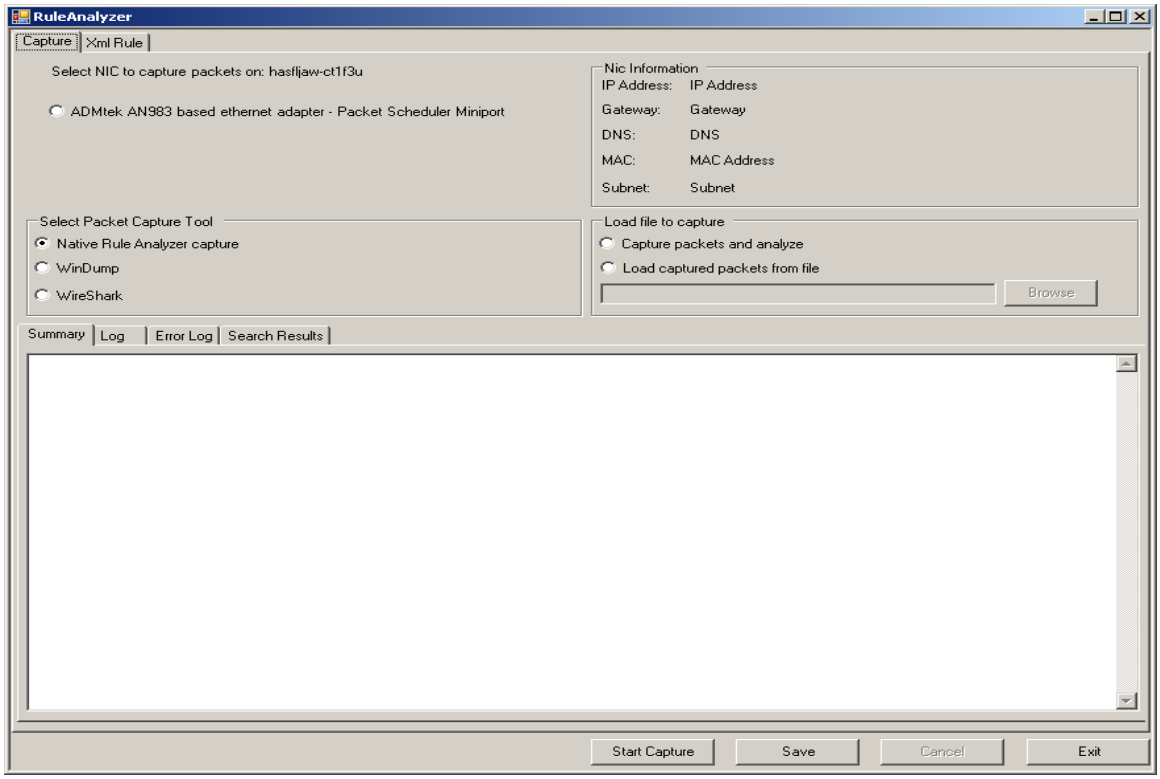


Figure 1

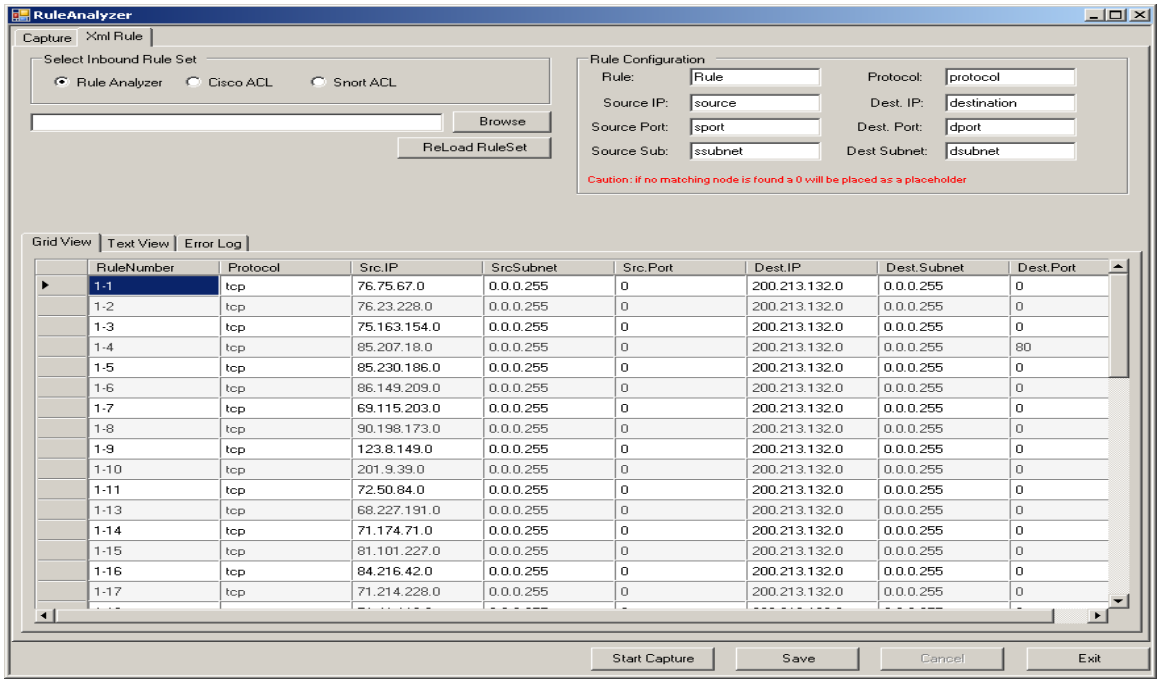


Figure 2

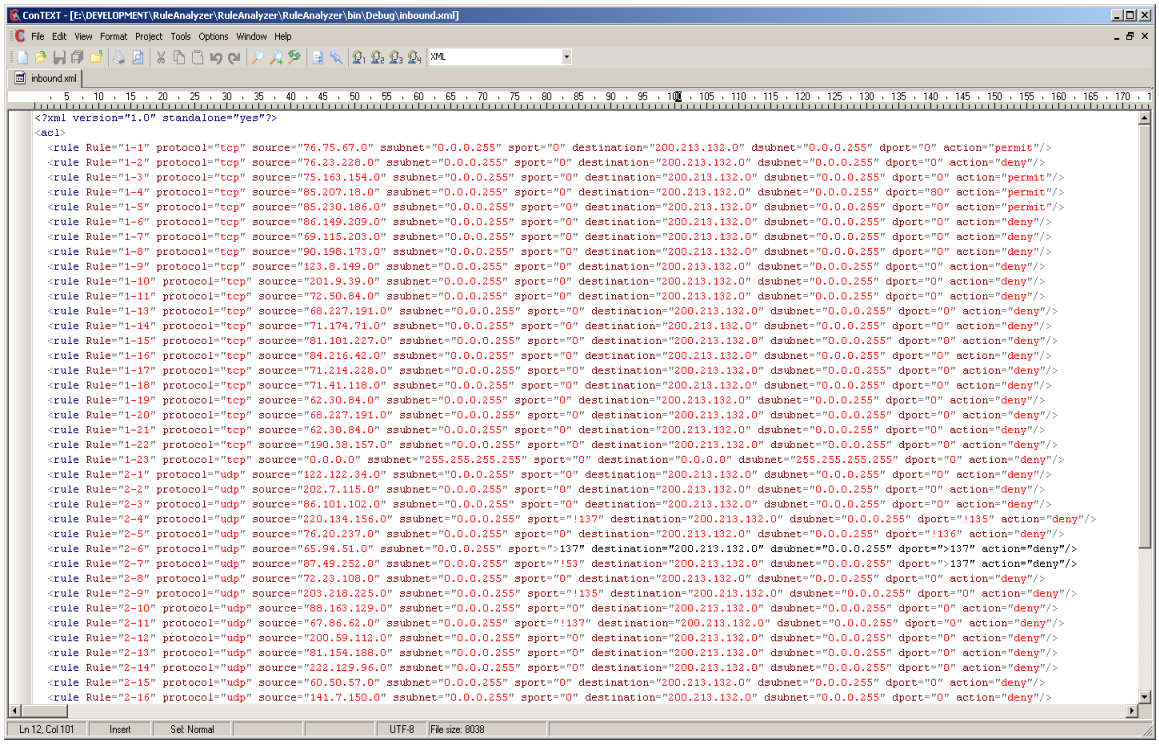


Figure 3

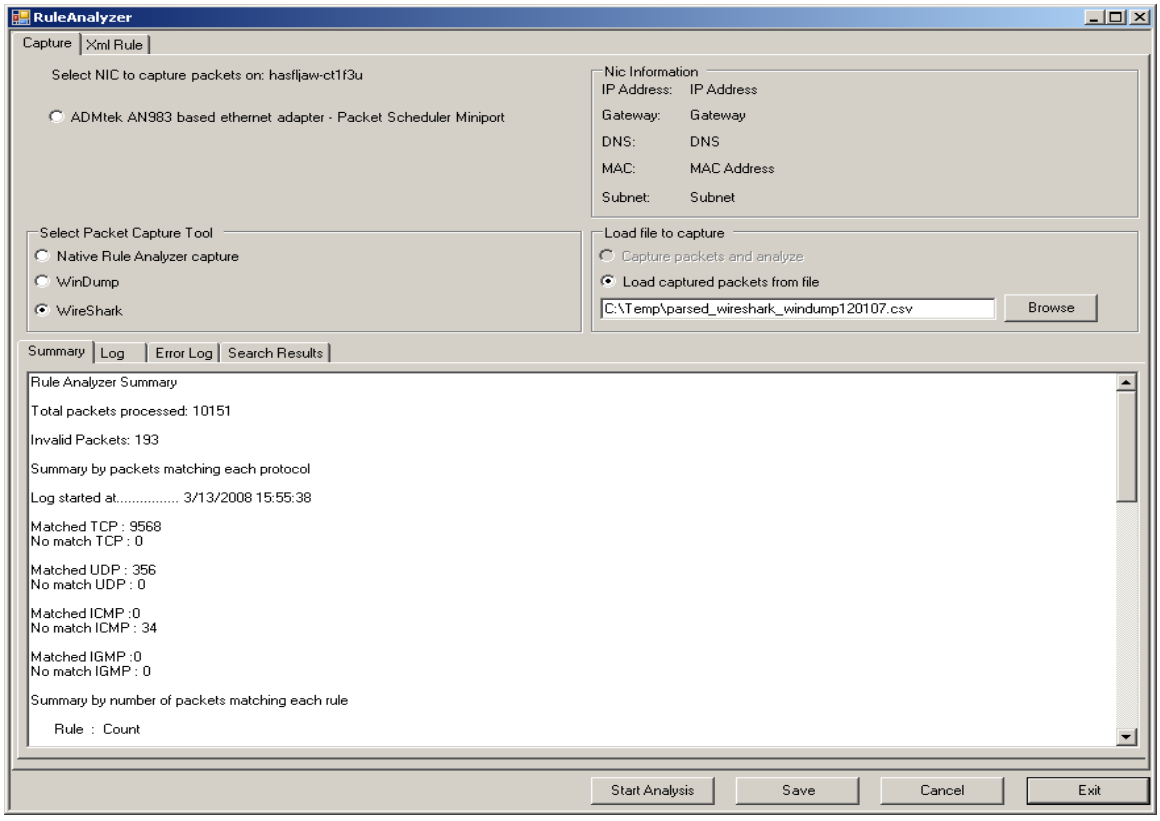


Figure 4

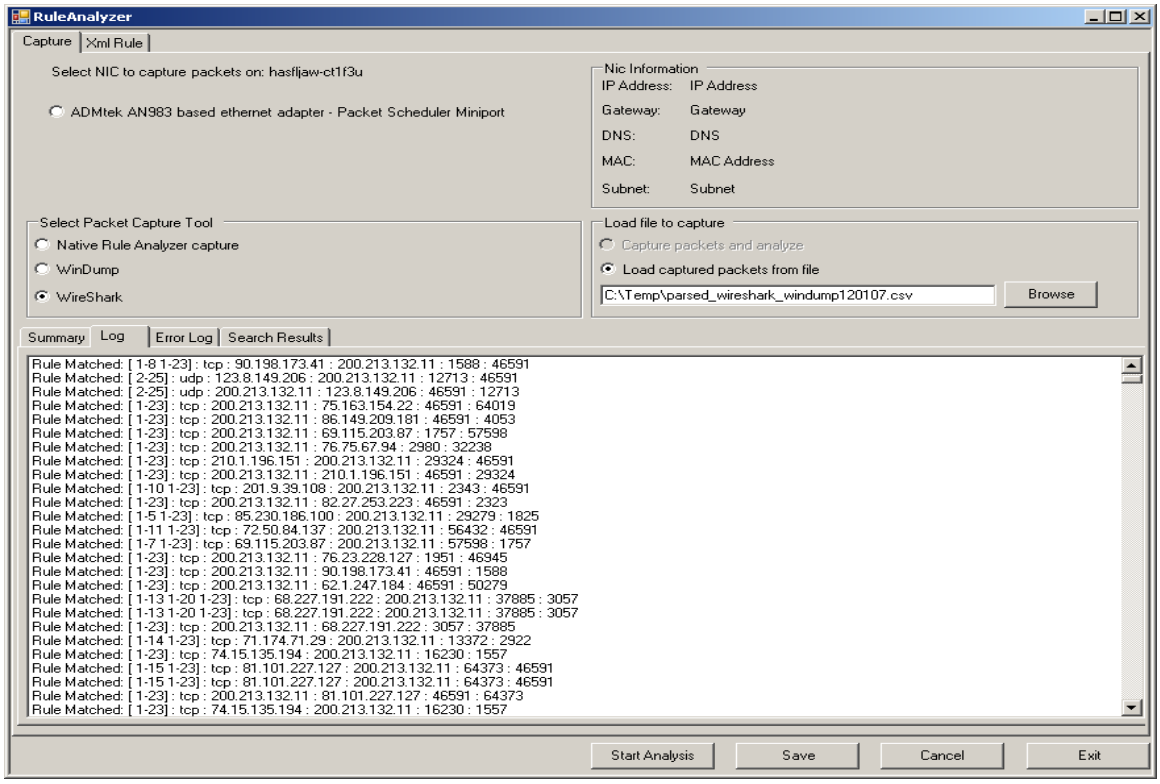


Figure 5

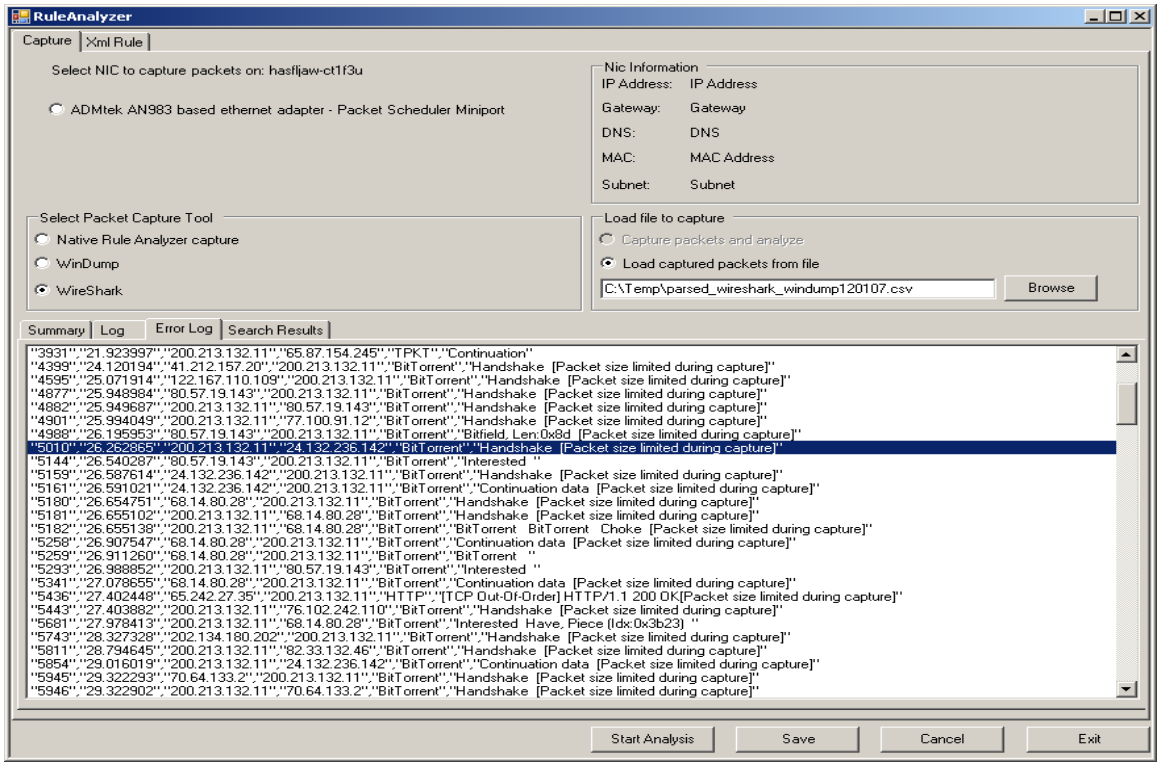
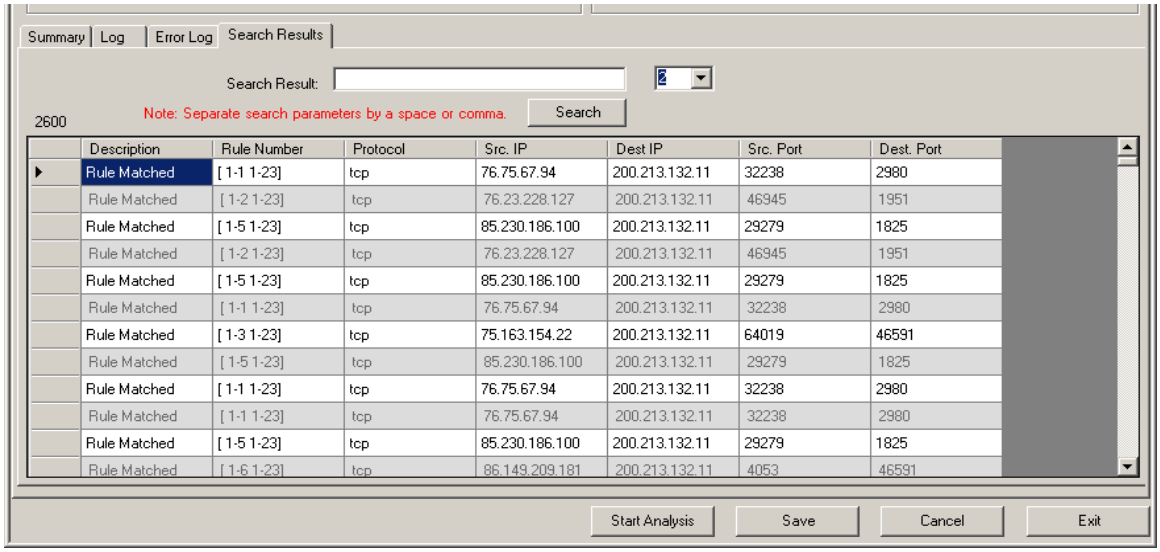


Figure 6



Description	Rule Number	Protocol	Src. IP	Dest IP	Src. Port	Dest. Port
Rule Matched	[1-1 1-23]	tcp	76.75.67.94	200.213.132.11	32238	2980
Rule Matched	[1-2 1-23]	tcp	76.23.228.127	200.213.132.11	46945	1951
Rule Matched	[1-5 1-23]	tcp	85.230.186.100	200.213.132.11	29279	1825
Rule Matched	[1-2 1-23]	tcp	76.23.228.127	200.213.132.11	46945	1951
Rule Matched	[1-5 1-23]	tcp	85.230.186.100	200.213.132.11	29279	1825
Rule Matched	[1-1 1-23]	tcp	76.75.67.94	200.213.132.11	32238	2980
Rule Matched	[1-3 1-23]	tcp	75.163.154.22	200.213.132.11	64019	46591
Rule Matched	[1-5 1-23]	tcp	85.230.186.100	200.213.132.11	29279	1825
Rule Matched	[1-1 1-23]	tcp	76.75.67.94	200.213.132.11	32238	2980
Rule Matched	[1-1 1-23]	tcp	76.75.67.94	200.213.132.11	32238	2980
Rule Matched	[1-5 1-23]	tcp	85.230.186.100	200.213.132.11	29279	1825
Rule Matched	[1-6 1-23]	tcp	86.149.209.181	200.213.132.11	4053	46591

Figure 7

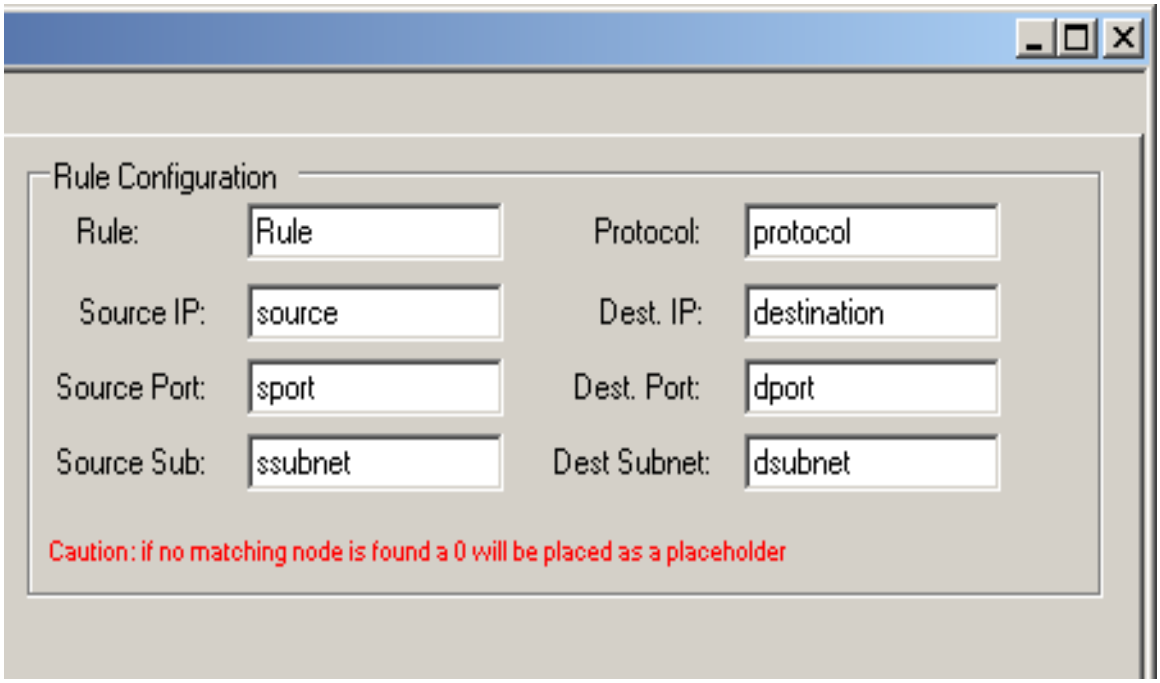


Figure 8

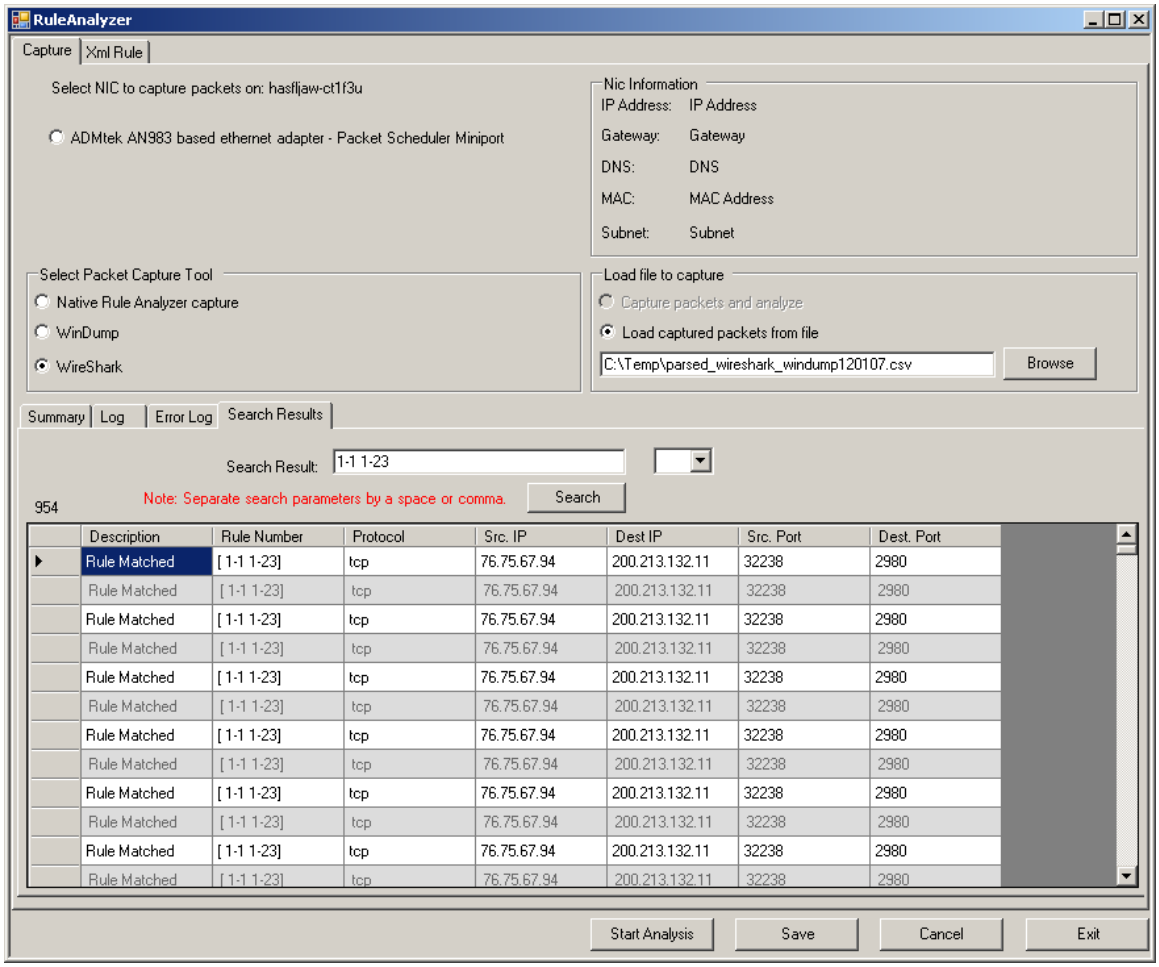

RuleAnalyzer
Capture
Xml Rule
Select NIC to capture packets on: hasfljaw-ct1f3u
ADMtek AN983 based ethernet adapter - Packet Scheduler Miniport
Nic Information
IP Address: IP Address
Gateway: Gateway
DNS: DNS
MAC: MAC Address
Subnet: Subnet
Select Packet Capture Tool
Native Rule Analyzer capture
WinDump
WireShark
Load file to capture
Capture packets and analyze
Load captured packets from file
C:\Temp\parsed_wireshark_windump120107.csv
Browse
Summary
Log
Error Log
Search Results
Search Result:
1-1 1-23
954
Note: Separate search parameters by a space or comma.
Search
Description
Rule Number
Protocol
Src. IP
Dest IP
Src. Port
Dest. Port
Rule Matched
[1-1 1-23]
tcp
76.75.67.94
200.213.132.11
32238
2980
Start Analysis
Save
Cancel
Exit

Figure 9

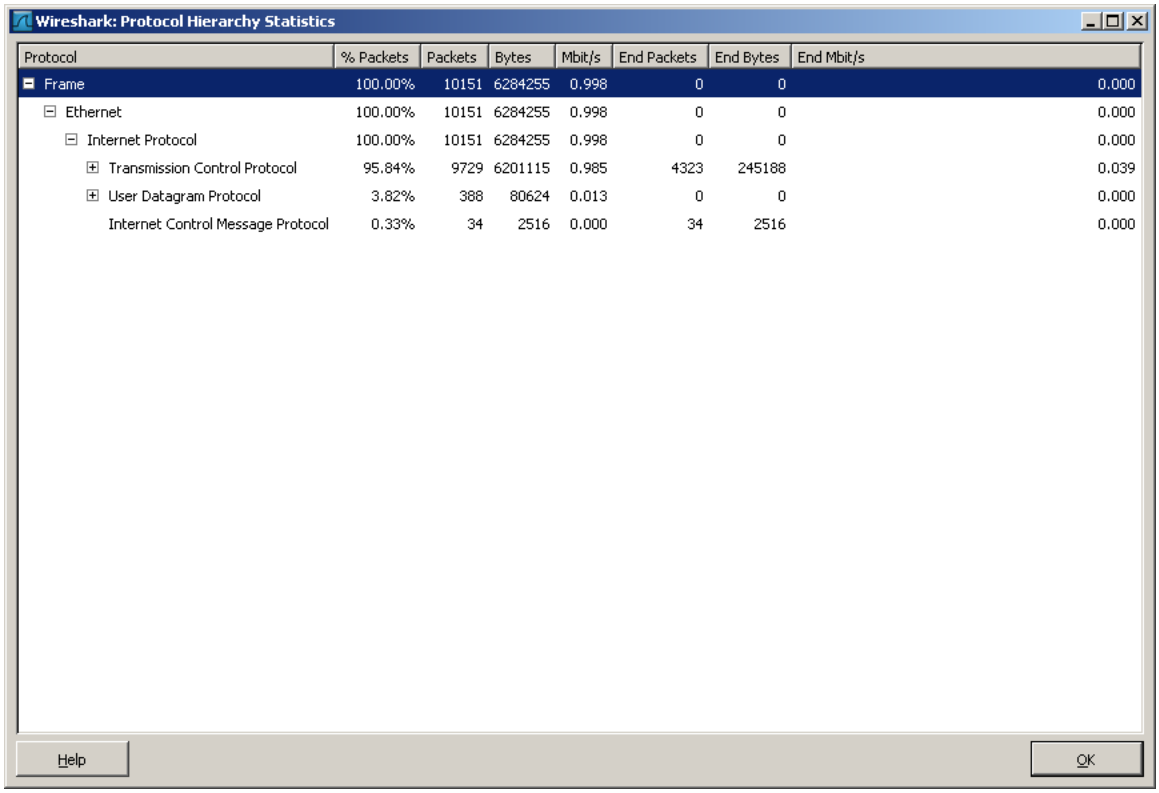

Wireshark: Protocol Hierarchy Statistics
Protocol
% Packets
Packets
Bytes
Mbit/s
End Packets
End Bytes
End Mbit/s
Frame 100.00% 10151 6284255 0.998 0 0 0.000
Ethernet 100.00% 10151 6284255 0.998 0 0 0.000
Internet Protocol 100.00% 10151 6284255 0.998 0 0 0.000
Transmission Control Protocol 95.84% 9729 6201115 0.985 4323 245188 0.039
User Datagram Protocol 3.82% 388 80624 0.013 0 0 0.000
Internet Control Message Protocol 0.33% 34 2516 0.000 34 2516 0.000
Help
OK

Figure 10

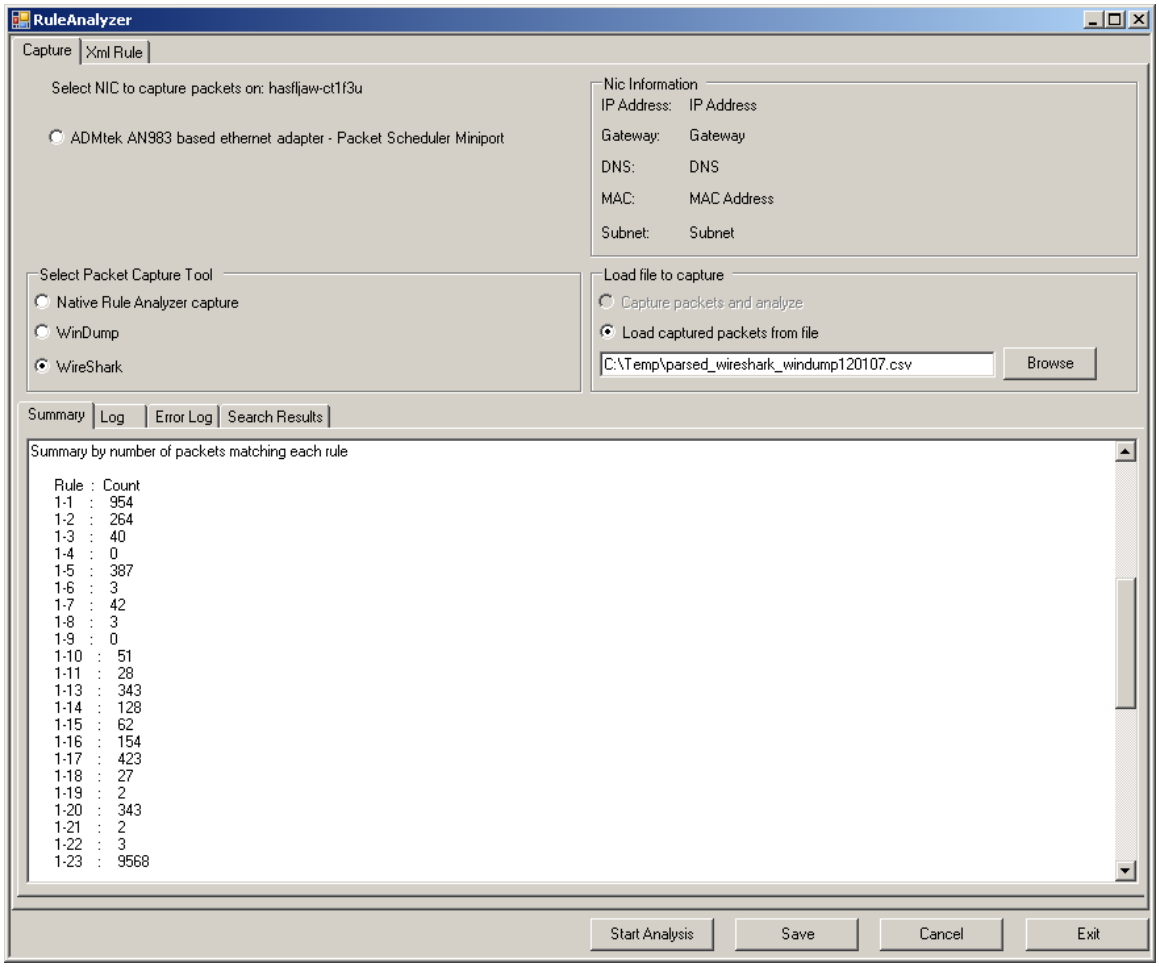


Figure 11

Sample Rules

Rule Analyzer XML Rules

```
<?xml version="1.0" standalone="yes" ?>
- <acl>
  <rule Rule="1-1" protocol="tcp" source="76.75.67.0" ssubnet="255.255.255.0" sport=">2900" destination="200.213.132.0" dsubnet="255.255.255.0" dport=">2900" action="permit" />
  <rule Rule="1-2" protocol="tcp" source="76.23.228.0" ssubnet="255.255.255.0" sport="0" destination="200.213.132.0" dsubnet="255.255.255.0" dport="2990" action="deny" />
  <rule Rule="1-3" protocol="tcp" source="75.163.154.0" ssubnet="255.255.255.0" sport="20:56000" destination="200.213.132.0" dsubnet="255.255.255.0" dport="20:56000" action="permit" />
  <rule Rule="1-4" protocol="tcp" source="85.207.18.0" ssubnet="255.255.255.0" sport="0" destination="200.213.132.0" dsubnet="255.255.255.0" dport="80" action="permit" />
  <rule Rule="1-5" protocol="tcp" source="85.230.186.0" ssubnet="255.255.255.0" sport="0" destination="200.213.132.0" dsubnet="255.255.255.0" dport="!80" action="permit" />
  <rule Rule="1-6" protocol="tcp" source="0.0.0.0" ssubnet="0.0.0.0" sport="0" destination="0.0.0.0" dsubnet="0.0.0.0" dport="0" action="deny" />
  <rule Rule="2-1" protocol="udp" source="122.122.34.0" ssubnet="255.255.255.0" sport="0" destination="200.213.132.0" dsubnet="255.255.255.0" dport="46000:47000" action="deny" />
  <rule Rule="2-2" protocol="udp" source="86.101.102.0" ssubnet="255.255.255.0" sport="0" destination="200.213.132.0" dsubnet="255.255.255.0" dport="0" action="deny" />
  <rule Rule="2-3" protocol="udp" source="76.20.237.0" ssubnet="255.255.255.0" sport="0" destination="200.213.132.0" dsubnet="255.255.255.0" dport="!136" action="deny" />
```

```
<rule Rule="2-4" protocol="udp" source="0.0.0.0" ssubnet="0.0.0.0" sport="0" destination="0.0.0.0"
   dsubnet="0.0.0.0" dport="0" action="deny" />
<rule Rule="3-1" protocol="icmp" source="0.0.0.0" ssubnet="0.0.0.0" sport="0" destination="0.0.0.0"
   dsubnet="0.0.0.0" dport="0" action="deny" />
 </acl>
```

Cisco Rules

access-list 101 permit tcp 76.75.67.0 255.255.255.0 gt 2900 200.213.132.0 255.255.255.0 gt 2900

access-list 102 permit tcp 76.23.228.0 255.255.255.0 200.213.132.0 255.255.255.0 eq 2990

access-list 103 permit tcp 75.163.154.0 255.255.255.0 range 20 56000 200.213.132.0 255.255.255.0 range 20 56000

access-list 104 permit tcp 85.207.18.0 255.255.255.0 200.213.132.0 255.255.255.0 eq 80 established

access-list 105 permit tcp 85.230.186.0 255.255.255.0 200.213.132.0 255.255.255.0 neq 80

access-list 106 deny tcp any any any any

access-list 107 permit udp 122.122.34.0 255.255.255.0 200.213.132.0 255.255.255.0 range 46000 47000

access-list 108 permit udp 86.101.102.0 255.255.255.0 200.213.132.0 255.255.255.0 any

access-list 109 permit udp 76.20.237.0 255.255.255.0 200.213.132.0 255.255.255.0 neq 136

access-list 110 deny udp any any any any

access-list 111 deny icmp any any

Snort Rules

1 tcp 76.75.67.0/22 >2900 -> 200.213.132.0/24 any (content: "|vert0 01 86 a5|"; msg:"mountd access";)

2 tcp 76.75.68.0/24 >2900 -> 200.213.132.0/24 any (content: "|vert0 01 86 a5|"; msg:"mountd access";)

3 tcp 76.75.74.0/24 20:56000 -> 200.213.132.0/24 20:56000 (content: "|vert0 01 86 a5|"; msg:"mountd access";)

4 tcp 85.207.18.0/24 20:56000 -> 200.213.132.0/24 80 (content: "|vert0 01 86 a5|"; msg:"mountd access";)

5 tcp 85.230.186.0/24 any -> 200.213.132.0/24 !80 (content: "|vert0 01 86 a5|"; msg:"mountd access";)

6 tcp any any -> any any (content: "|vert0 01 86 a5|"; msg:"mountd access";)

7 udp 122.122.34.0/24 any -> 200.213.132.0/24 46000:47000 (content: "|vert0 01 86 a5|"; msg:"mountd access";)

8 udp 86.101.102.0/24 any -> 200.213.132.0/24 any (content: "|vert0 01 86 a5|"; msg:"mountd access";)

9 udp 76.20.237.0/24 any -> 200.213.132.0/24 !136 (content: "|vert0 01 86 a5|"; msg:"mountd access";)

10 udp any any -> any any (content: "|vert0 01 86 a5|"; msg:"mountd access";)

11 icmp any any -> any any (content: "|vert0 01 86 a5|"; msg:"mountd access";)