

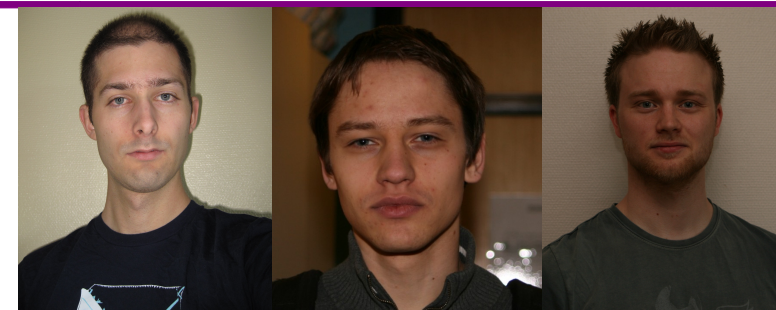
Master's thesis in  
Information- and Communication Technology

Title:

## Detecting malicious network activity using flow data and learning automata

Candidates:  
Christian Auby  
Torbjørn Skagestad  
Kristian Tveiten

Supervisor:  
Ole-Christoffer Granmo, UoA  
Morten Kråkvik, TSOC



### Problem

Malicious software is used by criminals to control computers connected to the internet, usually without the owner's knowledge. These applications are often used to collect sensitive information or perform illegal activities. Having such an application installed is a risk to home users as well as businesses.

One way to combat such a threat is to use an Intrusion detection System (IDS). There are several forms of IDS, but a common type is packet inspection systems (DPI). A DPI system works by monitoring a network connection, usually a mirror port on a switch. It analyses all the traffic that passes on this port looking for traffic generated by malware.

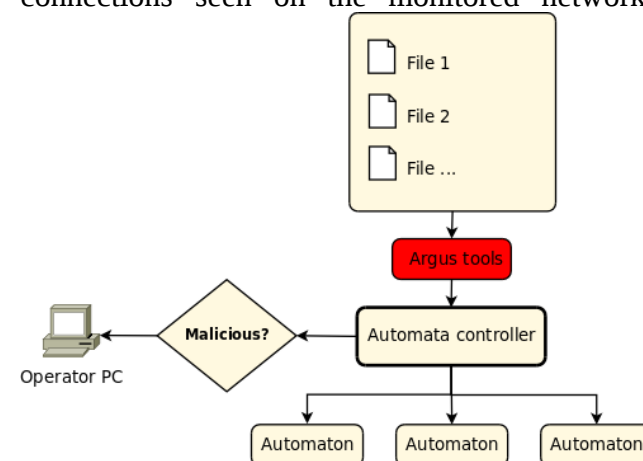
Analyzing the network content allows for complex rules that define malicious behavior. It also allows detection of many types of malware, but it comes at a price; computing resources. More bandwidth requires more computing resources, and computing resources require faster hardware through upgrades or load balancing using more than one IDS.

The problem we wanted to solve was to detect computers infected by malicious software at decreased resource requirements.

### Flow based IDS

Our proposed solution is a flow based analysis tool, implemented as a Python application. The existing tool Argus is used to capture flow data from the network and store them on disk.

The flow data stored by Argus consists of the IP-addresses, port number and data sizes of the connections seen on the monitored network.



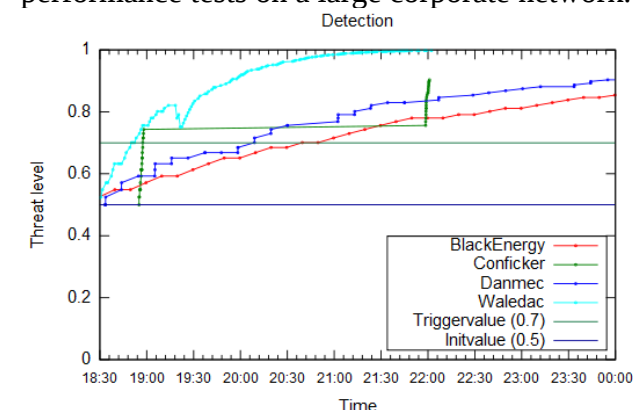
The figure shows the data path in our solution. The Argus files are read into our application which passes it to the automata controller. The controller sends the data to the correct automaton which provide feedback in return.

Each automaton looks at these values and builds a threat value for the source/destination pair it monitors using machine learning techniques.

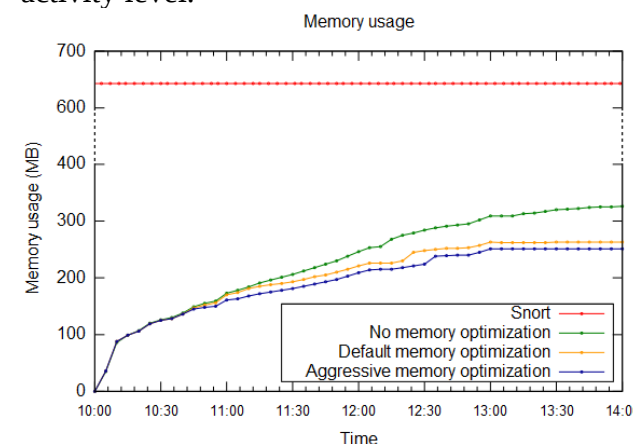
We have developed one automaton for each type of malware in our sample group, tuned to recognize the traffic generated by the respective sample.

### Experimental results

The detection tests were done using actual infected clients on real networks, and the performance tests on a large corporate network.



The graph shows detection over time for each malware sample that we included in our research. The different samples reach the trigger value at different times depending on their activity level.



Memory optimization was done by dropping old automata. This impacts memory usage as seen by the graph.

### Conclusion

We tested our solution in a large network that was able to produce over 100Mbit/s of traffic during work hours. The network consisted of approximately ten thousand workstations.

Looking at the detection results we are quite satisfied with the performance of our solution. The malware samples were detected in a reasonable amount of time. The amount of incorrect detections was low enough to easily find the actual infections.

The performance tests showed an average CPU usage of 10% compared to Snort which used 100%. Although our solution only detects a small set of infections the low resource usage makes it ideal for large networks.

Our solution is memory dependent. As the number of connection pairs increase the memory usage increases as well. This can be improved by various optimization techniques, as seen by the graph. With additional rules more optimization might be required depending on the network.

The load on the hardware during the selected period reduced Snort's ability to analyze packets because of packet loss. We believe our solution can be used to improve this ability by partly or completely exchanging Snort with a flow based solution.