Acta Informatica Pragensia
1(1), 2012, 50–59, DOI: 10.18267/j.aip.5
Online: aip.vse.cz

Section:
Peer-reviewed papers

Intrusion Detection Architecture Utilizing Graphics Processors

Liberios Vokorokos[1], Anton Baláž[1], Branislav Madoš[1]

[1] Department of Computers and Informatics,
Faculty of Electrical Engineering and Informatics, Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic

{liberios.vokorokos, anton.balaz, branislav.mados}@tuke.sk

Abstract: With the thriving technology and the great increase in the usage of computer networks, the risk of having these network to be under attacks have been increased. Number of techniques have been created and designed to help in detecting and/or preventing such attacks. One common technique is the use of Intrusion Detection Systems (IDS). Today, number of open sources and commercial IDS are available to match enterprises requirements. However, the performance of these systems is still the main concern. This paper examines perceptions of intrusion detection architecture implementation, resulting from the use of graphics processor. It discusses recent research activities, developments and problems of operating systems security. Some exploratory evidence is presented that shows capabilities of using graphical processors and intrusion detection systems. The focus is on how knowledge experienced throughout the graphics processor inclusion has played out in the design of intrusion detection architecture that is seen as an opportunity to strengthen research expertise.

Keywords: Attacks, Intrusion Detection Systems (IDS), Performance, Traffic, Suricata, GPU

1 INTRODUCTION

This work advocates intrusion detection architecture with the use of graphics processors, which is closely associated with the issue of computer systems security. Two areas are addressed in particular: design and implementation of the architecture.

As evidenced by an extant literature base, Intrusion Detection Systems (IDS) are now becoming one of the essential components in any organization's network, and are designed to detect any intrusion or hostile traffic in a network [1]. With the serious need of such detection systems organizations have been investing to produce a more effective IDS. However, the processing ability of IDS cannot catch tip with the rapid development of network bandwidth, which may give a rise to the safety risk.

In order to reduce this risk, it is necessary to analyse these data. However, this is a critical issue as common processors do not always provide enough power to do so [8].

IDS can be implemented as hardware- or software-based [1]. The later type of IDS is more configurable and easy to update while the hardware based is designed to handle large amount of traffic but more expensive and requires more maintenance as well. As guided by Zhang et al. [9], hardware (machines) based intrusion detection methods are increasingly being researched because they can detect unknown attacks. Their experiment results show that the time of reduction process can decrease significantly. That is, the new trend is drawing attention to the performance of graphics processors, which provide several times higher performance than the current processors.

The significant spare computational power and data parallelism capabilities of modern Graphics Processing Units (GPU) permit the efficient matching of multiple inputs at the same time against a large set of data [7]. Vasiliadis et al. demonstrate the feasibility of GPU regular expression matching by implementing it in the popular Snort intrusion detection system, which results to a 60% increase in the packet processing throughput [7].

The aims of this study are to gain further understanding on the most prominent aspects of an active safety system that relate to graphics processor performance. Thus, the goal of this work is to propose intrusion detection architecture with the use of graphics processors which can further lead to the provision of an active safety system for the particular network and operating system.

The findings show that the GPU can significantly contribute to acceleration of the network data analysis process. The paper closes with a look at the future for more research areas involving intrusion detection.

2 ANALYSIS

In order to design IDS with the use of a GPU, several programs and libraries ought to be applied and thus they need to be configured appropriately to the intrusion detection architecture.

In particular, Compute Unified Device Architecture (CUDA) has been applied to accomplish the use of graphic card performance with the IDS. In the CUDA programming model, the system consists of a host that is a traditional Central Processor Unit (CPU) and one or more massively data-parallel co-processing compute devices. A CUDA program consists of multiple phases executed on either the host or a compute device such as a GPU [4].

For intrusion detection, Suricata has been used, which is a rule-based IDS that takes advantage of externally developed rule sets to monitor sniffed network traffic and provide alerts when suspicious events take place [1]. Like most IDS, it is designed to fit within existing network security components. The initial release of Suricata runs on a Linux 2.6 platform and supports both inline and passive traffic monitoring configuration capable of handling multiple gigabit traffic levels [2]. Moreover, it supports multithreading and hardware acceleration as well.

The experimental architecture has been implemented on operating system Ubuntu 11.10 with the assumption that the graphic card is CUDA compatible.

Suricata has been configured with MySQL database, which has been used to save the outputs, and it has also been configured with Graphical User Interface (GUI) providing a better overview of the network traffic and its alerts. The GUI has been loaded through the web server, while Suricata has been compiled with CUDA libraries to enable support of the particular graphic card. That is, to analyze the network traffic, CPU has been used with the support of a GPU. Approximate scheme of the described architecture is depicted on the following Figure 1.

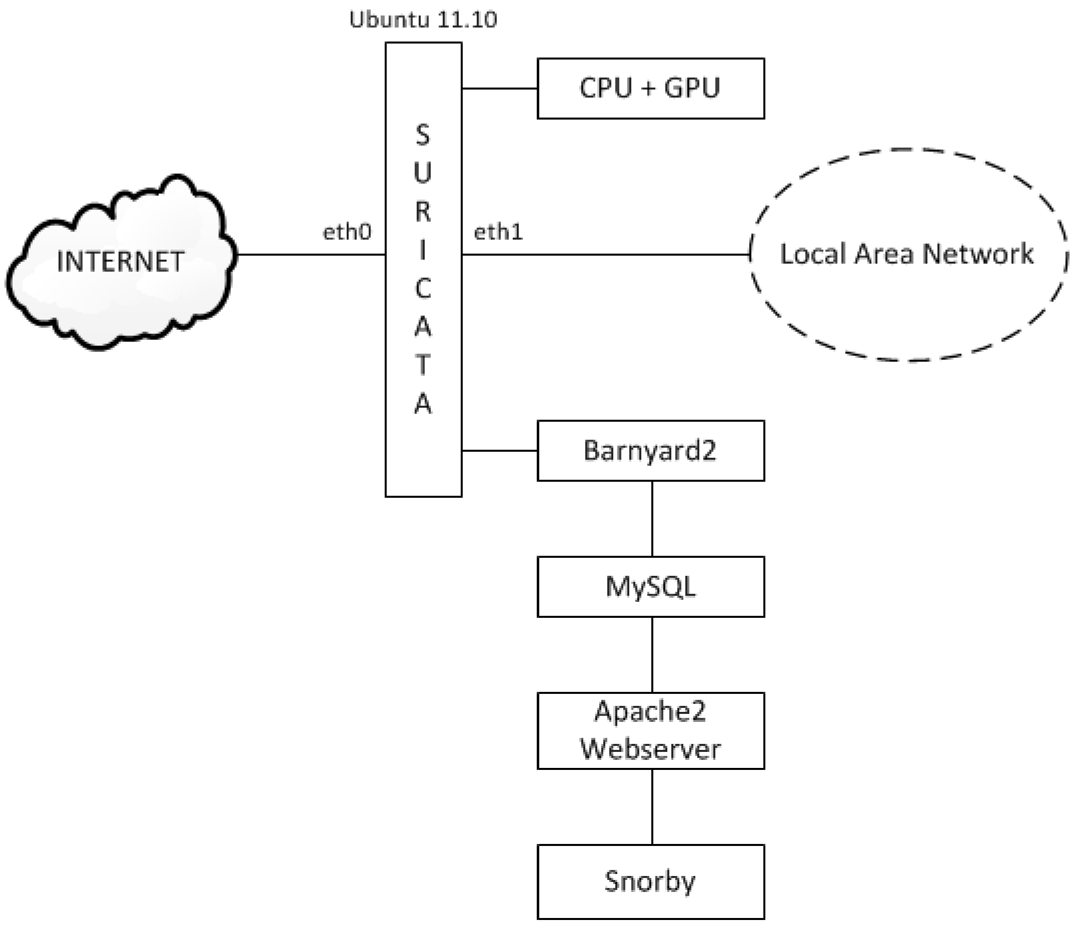
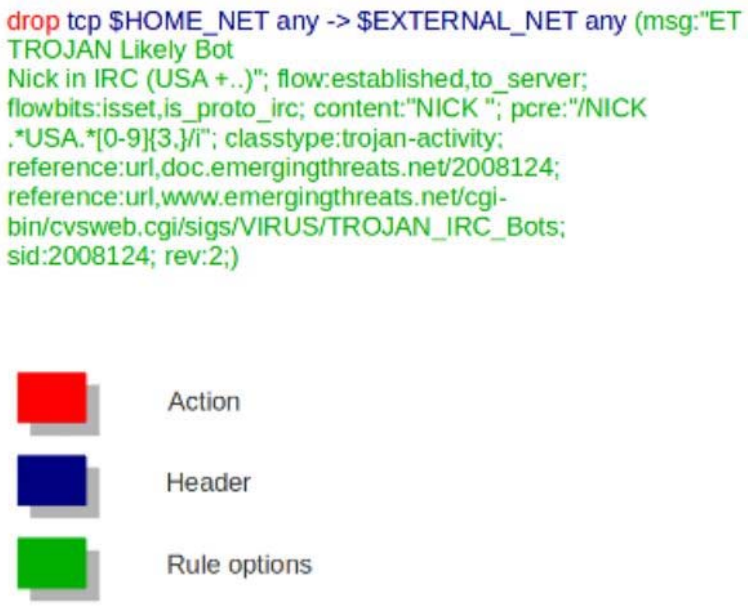


Fig. 1. Proposed Intrusion Detection Architecture

When a packet incomes to the network, it is collected via Packet Collector (PCAP). Then, selected rules are applied, that are online obtainable from Emerging Threats official website [3] and are easy manageable with the Rule Manager called Oinkmaster, which is a script that helps to update and manage Snort rules. It is released under the BSD license and works on most platforms that can run Perl scripts.

With the use of the selected rules, packets are analyzed with both GPU and CPU. There are two types of rules: Parallelizable and Non-parallelizable.

Generally, non-parallelizable rules are analyzed by the CPU and parallelizable rules are analyzed by the GPU on collected packet. If the GPU buffer is full, parallelizable rules are analyzed with the CPU as well. The following Figure 2 shows an example of a rule signature:



```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET
TROJAN Likely Bot
Nick in IRC (USA +..)"; flow:established,to_server;
flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; classtype:trojan-activity;
reference:url,doc.emergingthreats.net/2008124;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/VIRUS/TROJAN_IRC_Bots;
sid:2008124; rev:2;)
```

🟥 Action
🟦 Header
🟩 Rule options

Fig. 2. Rule Signature in Suricata

After packet analysis is performed, Suricata generates output which is processed and saved through the logging method called Barnyard to the MySQL database. Then, the results can be seen within the database, Suricata logs and the web server as well. Moreover, the web server also provides GUI, wherein statistical information about the network traffic can be found, e.g. alerts, alert priorities and graphs of the incidents.

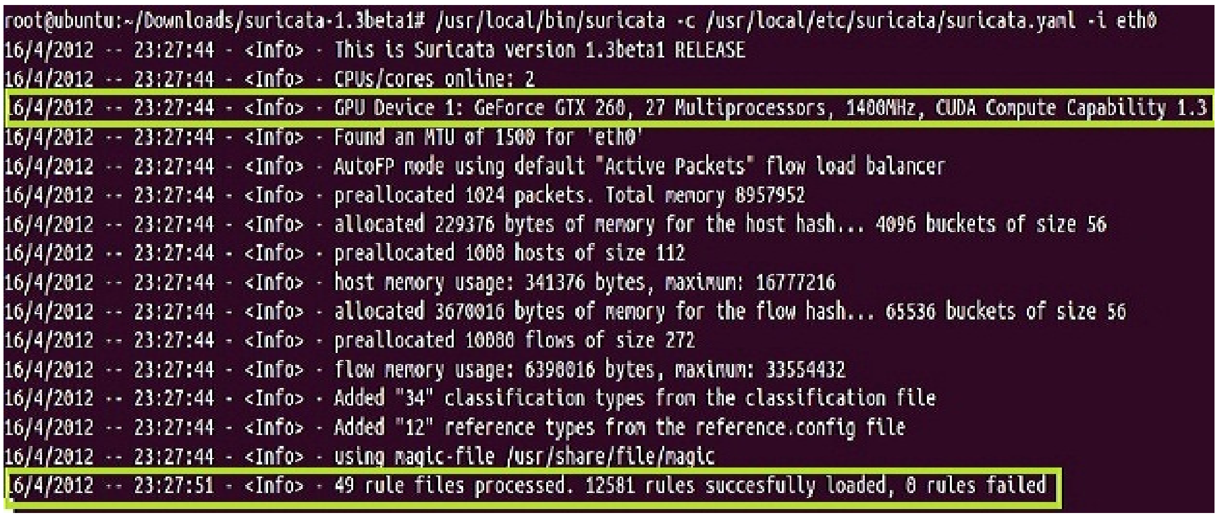
3 SOLUTION AND RESULTS

The main function of the architecture is to provide an active protection and raise the network security, which is achieved by IDS Suricata. The next function of the architecture is to use GPU and CPU for the network traffic analysis. The architecture also includes database, which is used for storage of the Suricata outputs. The GUI provides a better overview of the network traffic and alerts, which are generated by the IDS.

The architecture consists of the open-source software and libraries, providing the following functions:

1. IDS Suricata with the use of GPU and CPU
2. GUI Snorby (A Ruby-on-Rails web application intended for network security monitoring that interfaces with current popular IDS)
3. Ruleset for Suricata
4. Web server Apache2
5. MySQL database
6. CUDA architecture
7. TCPdump for collecting packets
8. Supporting Ruby-on-Rails applications for the web server using Phusion Passenger, a module for Apache2 web server
9. Generation of network traffic outputs via Barnyard

Suricata is compiled with the CUDA architecture and the `GeFORCE GTX 260` development drives. Emerging Threats rules are used, as they are free and up to date. The architecture also supports other rules, e.g. Snort VRT ruleset, which can be found at Snort official website [6]. The following Figure 3 depicts the launch of the IDS Suricata with both GPU and CPU, and 12581 rules.



```
root@ubuntu:~/Downloads/suricata-1.3beta1# /usr/local/bin/suricata -c /usr/local/etc/suricata/suricata.yaml -i eth0
16/4/2012 -- 23:27:44 - <Info> - This is Suricata version 1.3beta1 RELEASE
16/4/2012 -- 23:27:44 - <Info> - CPUs/cores online: 2
16/4/2012 -- 23:27:44 - <Info> - GPU Device 1: GeForce GTX 260, 27 Multiprocessors, 1400MHz, CUDA Compute Capability 1.3
16/4/2012 -- 23:27:44 - <Info> - Found an MTU of 1500 for 'eth0'
16/4/2012 -- 23:27:44 - <Info> - AutoFP mode using default "Active Packets" flow load balancer
16/4/2012 -- 23:27:44 - <Info> - preallocated 1024 packets. Total memory 8957952
16/4/2012 -- 23:27:44 - <Info> - allocated 229376 bytes of memory for the host hash... 4096 buckets of size 56
16/4/2012 -- 23:27:44 - <Info> - preallocated 1000 hosts of size 112
16/4/2012 -- 23:27:44 - <Info> - host memory usage: 341376 bytes, maximum: 16777216
16/4/2012 -- 23:27:44 - <Info> - allocated 3670016 bytes of memory for the flow hash... 65536 buckets of size 56
16/4/2012 -- 23:27:44 - <Info> - preallocated 10000 flows of size 272
16/4/2012 -- 23:27:44 - <Info> - flow memory usage: 6390016 bytes, maximum: 33554432
16/4/2012 -- 23:27:44 - <Info> - Added "34" classification types from the classification file
16/4/2012 -- 23:27:44 - <Info> - Added "12" reference types from the reference.config file
16/4/2012 -- 23:27:44 - <Info> - using magic-file /usr/share/file/magic
16/4/2012 -- 23:27:51 - <Info> - 49 rule files processed. 12581 rules succesfully loaded, 0 rules failed
```

Fig. 3.IDS Implementation Using GPU and CPU

Suricata provides multiple run modes, each of which initializes the threads, queues, and plumbing necessary for operation. These modes are usually tied to the choice of a capture device and whether the mode is IDS or IPS (Intrusion Prevention System). The PCAP has been used in daemon mode and TCPdump for capturing files for testing GPU vs. CPU. Only one run mode is chosen at startup.

The process of how the packets are captured is shown in Figure 4. Modules are used to encapsulate a single primary function with lifecycle callbacks. Each thread in the packet pipeline represents an instance of a module. These threads are initialized by the runmode defined in the source file `runmodes.c`. The runmode also initializes the queues and packet handlers which are used for moving packets between the modules and queues. A thread is marked as `unnable` after all the steps from the runmode initialization are complete.

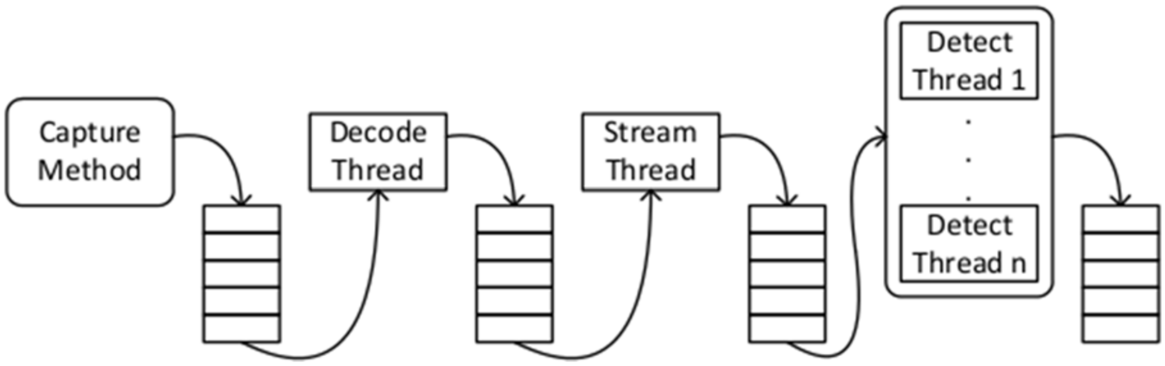


Fig. 4. Packet Pipeline

The `pcap` device is initialized using the provided name, e.g. `eth0`. Once a device is initialized it will begin with gathering the packets and passing them to Suricata. Suricata then acts as a thin wrapper around the data provided, making it compatible with the link type decoders.

Decoding is the process of taking a buffer and converting its content to a Suricata support data structure. These buffers are handed off to a specific link type decoder.

The detection module takes care of multiple complex tasks: Loading all signatures, initializing detection plugins, creating detection groups for packet routing, and finally running packets through all applicable rules.

Figure 5 illustrates GUI configured and running with Suricata, with events and statistics of the network traffic. It uses web server Apache2 with which the cooperation is reached at the hand of the Ruby-on-Rails gems and the Phusion Passenger. The GUI uses results from the MySQL database and Suricata logs.

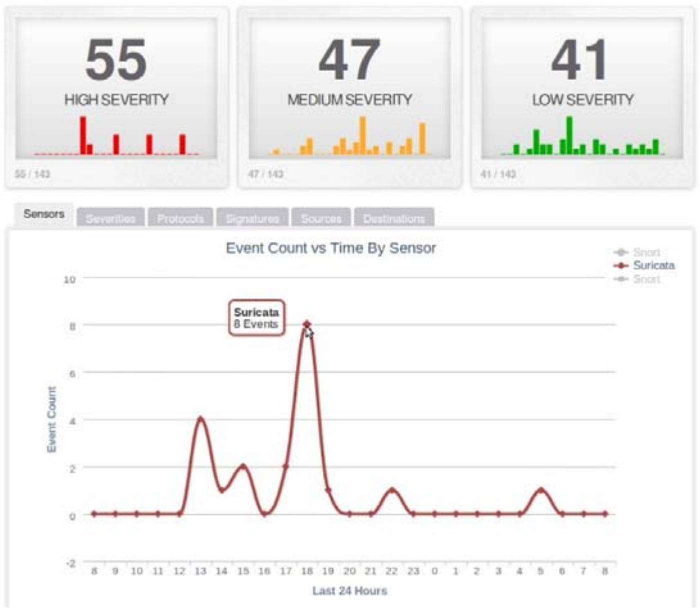


Fig. 5. Graphical User Interface

4 CONCLUSION

Network analysis involves processing large amounts of data to look for those that are suspicious. This is time consuming and requires a large amount of processing power, often affecting other running processes. By using hardware included in most personal computers, a performance can be seen. The processing of files can be done on a Graphics Processing Unit (GPU), contained in common video cards. A GPU is perfect for this because of its strong similarities to a Central Processing Unit (CPU).

The aims of this work were to design an IDS architecture that would be capable of the use of the GPU. That is, the work intended to propose an IDS architecture cooperating with the graphics processors which can further lead to the provision of an active safety system for the particular network and operating system.

Since the GPU has multiple processing units (32–128), it has an advantage over a CPU (1–8 processing units) [5]. This allows a single data stream to be processed using different metrics in parallel, while consuming minimal clock cycles from the CPU. A GPU also has its own memory, separated from the CPU, but also has the ability to share part of the CPU's memory.

By using the GPU, other processes do not experience performance decreases by fighting over CPU usage. Using a GPU can also be applied to intrusion detection systems (IDS) and firewall based applications, e.g. in addition to anti-malware applications.

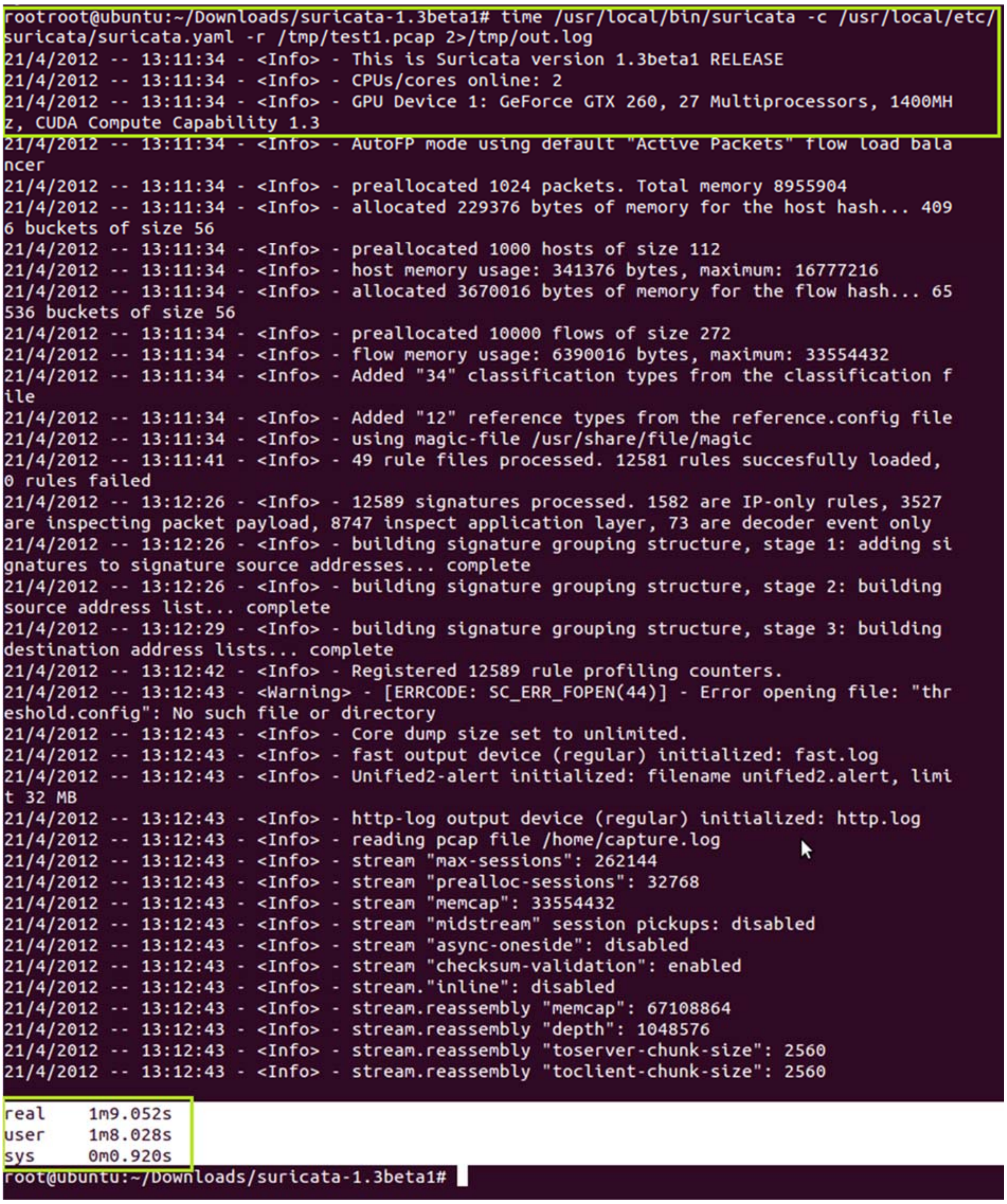
This research in progress investigated the use of GPUs for monitoring or detecting malicious activity on a network. Specifically, we compared fully functional architecture running with and without the GPU. We also analyzed the same captured output with size of 20 MB over 12581 rules. With support of the `GPU NVIDIA GTX 260`, the time of the test was 1 minute and 9 seconds. Without the GPU, the time was 1 minute and 28 seconds, from which we can conclude that the detection using the GPU was 19 second faster. The results of the test with the GPU are depicted in the Figure 6.

One advantage is that the open-source software was applied. Moreover, the latest versions of operating system, software, libraries, and rules were used, eliminating security risks.

On the other hand, the architecture does not support `SLi` which is used for parallel computing with more graphics cards at the same time. Further, the architecture does not support `ATI` cards and if `NVIDIA` card is used, it must be CUDA compatible.

Although extant research has highlighted the importance of considering both CPU and GPU as means to accelerate network data analysis, the present study added important experiments of detail to this research by distinguishing between an increase in the IDS response times and a decrease in the time required to analyze network data.

Although past studies have focused on the factor structure of GPU (e.g. [5]) and have investigated the effects that may account for differences in the use of various IDS (e.g. [1]), the present study is an important extension of this work as well.



```
rootroot@ubuntu:~/Downloads/suricata-1.3beta1# time /usr/local/bin/suricata -c /usr/local/etc/
suricata/suricata.yaml -r /tmp/test1.pcap 2>/tmp/out.log
21/4/2012 -- 13:11:34 - <Info> - This is Suricata version 1.3beta1 RELEASE
21/4/2012 -- 13:11:34 - <Info> - CPUs/cores online: 2
21/4/2012 -- 13:11:34 - <Info> - GPU Device 1: GeForce GTX 260, 27 Multiprocessors, 1400MH
z, CUDA Compute Capability 1.3
21/4/2012 -- 13:11:34 - <Info> - AutoFP mode using default "Active Packets" flow load bala
ncer
21/4/2012 -- 13:11:34 - <Info> - preallocated 1024 packets. Total memory 8955904
21/4/2012 -- 13:11:34 - <Info> - allocated 229376 bytes of memory for the host hash... 409
6 buckets of size 56
21/4/2012 -- 13:11:34 - <Info> - preallocated 1000 hosts of size 112
21/4/2012 -- 13:11:34 - <Info> - host memory usage: 341376 bytes, maximum: 16777216
21/4/2012 -- 13:11:34 - <Info> - allocated 3670016 bytes of memory for the flow hash... 65
536 buckets of size 56
21/4/2012 -- 13:11:34 - <Info> - preallocated 10000 flows of size 272
21/4/2012 -- 13:11:34 - <Info> - flow memory usage: 6390016 bytes, maximum: 33554432
21/4/2012 -- 13:11:34 - <Info> - Added "34" classification types from the classification f
ile
21/4/2012 -- 13:11:34 - <Info> - Added "12" reference types from the reference.config file
21/4/2012 -- 13:11:34 - <Info> - using magic-file /usr/share/file/magic
21/4/2012 -- 13:11:41 - <Info> - 49 rule files processed. 12581 rules succesfully loaded,
0 rules failed
21/4/2012 -- 13:12:26 - <Info> - 12589 signatures processed. 1582 are IP-only rules, 3527
are inspecting packet payload, 8747 inspect application layer, 73 are decoder event only
21/4/2012 -- 13:12:26 - <Info> - building signature grouping structure, stage 1: adding si
gnatures to signature source addresses... complete
21/4/2012 -- 13:12:26 - <Info> - building signature grouping structure, stage 2: building
source address list... complete
21/4/2012 -- 13:12:29 - <Info> - building signature grouping structure, stage 3: building
destination address lists... complete
21/4/2012 -- 13:12:42 - <Info> - Registered 12589 rule profiling counters.
21/4/2012 -- 13:12:43 - <Warning> - [ERRCODE: SC_ERR_FOPEN(44)] - Error opening file: "thr
eshold.config": No such file or directory
21/4/2012 -- 13:12:43 - <Info> - Core dump size set to unlimited.
21/4/2012 -- 13:12:43 - <Info> - fast output device (regular) initialized: fast.log
21/4/2012 -- 13:12:43 - <Info> - Unified2-alert initialized: filename unified2.alert, limi
t 32 MB
21/4/2012 -- 13:12:43 - <Info> - http-log output device (regular) initialized: http.log
21/4/2012 -- 13:12:43 - <Info> - reading pcap file /home/capture.log
21/4/2012 -- 13:12:43 - <Info> - stream "max-sessions": 262144
21/4/2012 -- 13:12:43 - <Info> - stream "prealloc-sessions": 32768
21/4/2012 -- 13:12:43 - <Info> - stream "memcap": 33554432
21/4/2012 -- 13:12:43 - <Info> - stream "midstream" session pickups: disabled
21/4/2012 -- 13:12:43 - <Info> - stream "async-oneside": disabled
21/4/2012 -- 13:12:43 - <Info> - stream "checksum-validation": enabled
21/4/2012 -- 13:12:43 - <Info> - stream."inline": disabled
21/4/2012 -- 13:12:43 - <Info> - stream.reassembly "memcap": 67108864
21/4/2012 -- 13:12:43 - <Info> - stream.reassembly "depth": 1048576
21/4/2012 -- 13:12:43 - <Info> - stream.reassembly "toserver-chunk-size": 2560
21/4/2012 -- 13:12:43 - <Info> - stream.reassembly "toclient-chunk-size": 2560

real    1m9.052s
user    1m8.028s
sys     0m0.920s
root@ubuntu:~/Downloads/suricata-1.3beta1#
```

Fig. 6. Results of the Test with the GPU

Additional research should be performed in the incorporation of signature and anomaly-based intrusion detection to meet the unknown and persistent threats to information and data infrastructure. That is, future research should address the integration of both signature and anomaly-based intrusion detection into a unified and seamless solution.

ACKNOWLEDGEMENT:

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0008-10.

REFERENCES

[1] ALHOMOUD, A., MUNIR, R., DISSO, J.P., AWAN, I., AL-DHELAAN, A. *Procedia Computer Science,* Volume 5*: Performance Evaluation Study of Intrusion Detection Systems*, pp. 173–180, 2011.

[2] DAY, D., BURNS, B. *Proceedings of the 5th International Conference on Digital Society: A Performance Analysis of Snort and Suricata NetworkIntrusion Detection and Prevention Engines*, pp. 187–192, 2011, ISBN 978-1-61208-116-8.

[3] *Emerging Threats: Rule Documentation*, 2011, URL http://doc.emergingthreats.net/

[4] KOLLER, T. *Compute Unified Device Architecture (CUDA)*, 2012, URL http://ps.fb12.uni-siegen.de/downloads/Seminare/multicore-ws2011/koller.pdf/

[5] KOVACH, N., MULLINS, B.*Proceedings of the 5th International Conference on Information Warfare and Security: Malware Detection via a Graphics Processing Unit*, Wright Patterson AFB, Ohio, pp. 212–215, 2010.

[6] *Snort Official Website*, 2012, URL http://www.snort.org/

[7] VASILIADIS, G., POLYCHRONAKIS, M., ANTONATOS, S., MARKATOS, E., IOANNIDIS, S. *Recent Advances in Intrusion Detection*, Volume 5758: *Regular Expression Matching on Graphics Hardware for Intrusion Detection*, St Malo, France, pp. 265–283, 2009.

[8] WU C., TIN, J., CAI, Z., ZHU, E., CHENG, J. *Security Technology*, Volume 58: *An Efficient Pre-filtering Mechanism for Parallel Intrusion Detection Based on Many-Core GPU*, Jeju Isl, South Korea, pp. 298–305, 2009.

[9] ZHANG, X., ZHAO, C., WU, J., SONG, C. *Theoretical and Mathematical Foundations of Computer Science*, Volume 164: *A GPU-RSVM Based Intrusion Detection Classifier*, Singapore, Malaysia, pp. 92–100, 2011.