# An Approach for Unifying Rule Based Deep Packet Inspection

A. Munoz, S. Sezer, D. Burns, G Douglas
Centre for Secure Information Technologies (CSIT),
Queen's University of Belfast, Belfast, Northern Ireland, UK

**Abstract—High performance Internet traffic inspection and layer-7 content analysis have become essential functions of high speed networks. Over the past decade several DPI systems have evolved targeting specific issues related to traffic management, user/application policing, intrusion detection/prevention, URL/malicious/unwanted content filtering. Snort, OpenDPI, Bro, L7-filter, ClamAV are a number of open-source tools based on custom DPI engines and custom rule-sets. The surging demand for higher bandwidth DPI systems capable of supporting larger rule-sets requires the use of hardware acceleration and hardware-based systems. In comparison to software based systems, the design and development of custom purpose hardware for DPI is expensive and enforces the need for a unified DPI system that can be used for a wide range of DPI applications.**

**This paper presents the research in converting known DPI rule-sets into a meta format based on regular expression, that can be executed by a software and hardware-based processing platforms. To demonstrate this work a Snort2Regex translator has been developed to transform Snort rules into regular expressions using not only the content of the Snort rule but every relevant element that belongs to it and could increase the accuracy of the analysis.**

## 1. Introduction

The Internet is the baseline communication technology for a range of services including education, health, entertainment and ecommerce. As a global infrastructure, the Internet has also become a medium and space for organized and opportunistic crimes. Initial security threats have evolved and now range from harmless virus attacks to cyber-crime and cyber-war, incentivized by significant financial and political gains and objectives.

In order to ensure a safe and secure use of the Internet, user/application policing, network security and Quality of Service (QoS) systems need high performance traffic and content analysis capability. These methods of analysis are also known as Deep Packet Inspection (DPI).

The principles of packet content inspection evolved with the need for an effective network security solution, when traditional firewall technology, based on TCP/IP header fields, began to fail. High performance Internet traffic inspection and layer-7 content analysis become essential functions of high speed networks. Over the past decade several DPI systems have evolved targeting specific issues related to traffic management, user/application policing, intrusion detection/prevention, URL/malicious/unwanted content filtering. SNORT, OpenDPI, Bro, L7-filter, ClamAV are a number of open-source tools based on custom DPI engine and custom rule-sets.

One of the most used signature collections belongs to Snort [1]. Snort is an open source network intrusion prevention and detection system (IDS/IPS). Initially developed by Martin Roesch, Snort is now developed and maintained by SourceFire [2]. Integrated enterprise versions with purpose built hardware and commercial support services are commercialized by SourceFire.

Based on open community support, Snort rules are widely used for security purposes. They are however only tailored for use with the Snort IDS and cannot be used with any other systems in their native format. The open nature of the system makes it easier to develop new rules and integrate them into Snort. Companies continuously develop their own rules to support deficiencies of software and hardware against malicious activities.

Over the past 10 years the user link speed has increased by 35 times. Cisco forecasts a fourfold increase in web traffic by 2014, to 63.9 Exabytes per month [3]. The performance gap between processing requirements of DPI systems and their software-based implementation has been increasing in the past years mainly due to the increased connection speeds, data volume and rule database sizes. Hardware acceleration and hardware based DPI systems are the most promising solutions to close this technology gap. In comparison to software based systems, the design and development of custom purpose hardware for DPI is expensive and enforces the need for a unified DPI system that can be used for a wide range of DPI applications.

This paper presents the research and development of a DPI meta-rule-format based on regular expressions that can be executed by software and hardware-based processing platforms. The meta-rule-format provides the necessary expression richness to support most known DPI rules including Snort, Bro, L7-filter, ClamAV and can be efficiently translated and executed by a single DPI platform.

This paper targets mainly the translation methods for transforming Snort rules into regular expressions using not only the content but also the header of the Snort rules. The remainder of this paper is organized as follows: Section 2 briefly introduces related work concerning DPI Systems and rule signature collections. Section 3 introduce the motivation. Section 4 presents the methodology of the transformation and describes the Snort keywords structure. Section 5 shows the results obtained by this method. Section 6 presents conclusions and the future work.

## 2. RELATED WORK

There are a number of published papers describing methods that can be used to convert signature syntax from one specification to another. These methods allow well renowned signature collections to be used by DPI systems other than those which they were designed for, taking advantage of their accuracy and versatility.

R. Sommer and V. Paxson [4] creates Snort2Bro that converts Snort signatures into Bro signatures. Bro intrusion detection system [5] was originally written by Vern Paxson at Lawrence Berkeley National Lab and the International Computer Science Institute. Bro collects, filters and analyzes traffic that passes through a specific network location against a collection of Bro policy scripts [6]. "Bro policy scripts" are programs written in Bro's language and contain the rules that monitor, analyze and initiate actions based on the result. This strongly typed language allows users to detect typing inconsistencies at compile-time, and guarantees that all variable references at runtime will be set to valid values. One of the handicaps Bro faces against Snort is the lack of an up-to-date set of signatures. You can either create your own signatures for specific attacks using Bro's signature syntax, or you can try to port other security systems rules.

Bro policies also incorporate signature matching for DPI monitoring, these signatures are expressed as regular expressions. But while the policy script is capable of performing traditional signature-matching, large sets of signatures can be hard to use because each signature has to be coded as part of a script function. In addition, if the signatures are matched sequentially, then the overhead of the matching can become prohibitive.

Steven T. Eckmann [7] designed a system called Snort2Statl to translate Snort rules to STATL scenarios. STAT (State Transition Analysis Technique) is a framework developed by the Reliable Software Group at UCSB for building IDS systems [8], it includes an extensible description language for state/transition-based attack called STATL [9] designed to support intrusion detection. This language helps to describe the sequence of actions performed during a system attack.

Initially STAT was used as the basic platform for a host-based intrusion detection system called USTAT development [10] and extended to a network-based intrusion detection systems called NetSTAT [11]. The evaluation of these tools lead to the STAT redesign as a family of systems based on STATL and the domain-independent characteristics of the STAT approach embedded in a runtime called "core". The STAT framework has been use as a base platform to develop a number of intrusion detection systems for different environments, e.g. WinSTAT for Windows NT event logs or WebSTAT for Apache web server logs.

A. Tongaonkar, S. Vasudevan and R. Sekar [12] improve the performance of a packet classification system by using a native compilation of rules on a high-level specification language with a special type-system for packet processing. The C code generated can be compiled using a common C compiler. The native code generated can be loaded into Snort as a shared library to increase the performance of the system, their results show an improvement up to a factor of five in Snort packet classification.

## 3. MOTIVATION

Although previous work had addressed similar transformations, none of them use the entire Snort rule set for a regular expression transformation. The method proposed in [4] has not been maintained since the publication of the paper and today it is clearly outdated. Since Snort has evolved and expanded its rule options to take advantage of certain pre-processors specific functionality, these options cannot be ported to Bro without hindering the detection range. The method proposed by S. T. Eckmann at [7] addresses the rule header and the rule options of Snort to translate the rules into STATL description language for NetSTAT. Their method however can't be used to derive an abstract language that could be used among other systems. The approach followed at [12] presents a similar handicap. While we believe this method could be used by other C-based DPI systems and the C language is widely used in software programming, only those systems could take advantage of this method. FPGA and NPU based DPI hardware systems will then require custom solutions.

There is a need to share signature collections among different systems. Rules created to protect users or detect applications should be shared independently of the platform. Only an abstract language manageable by hardware and software DPI implementations could allow specialist rules writers to focus on the treats not on the platform. This paper presents a method that uses standard POSIX syntax for regular expressions to define security rules. The syntax proposed make it possible to perform DPI analysis with a database of rules for different hardware and software DPI pattern matching engines.

One of the benefits of regular expression syntax is that the versatility of the language offers the possibility to rewrite a regular expression in multiple forms with different styles. The advantage of this approach is that it could be adapted in many ways. Adjusting the rule to target specific DPI systems means performance could be improved by focusing on the strong points of the system capabilities.

## 4. METHODOLOGY

Snort2Regex is the implementation of a parser that reads Snort rules and extracts the different fields that compose the rule classifying them. After the different fields are interpreted and transformed into hexadecimal values (if it is necessary) a regular expression will be created with all the fields tailored to match an internet packet. That regular expression will match a internet packet in the same way the Snort rule will. Snort2Regex is based on Snort version 2.8.5, the current version implements the vast majority of keywords and function included in this version of Snort.

The Snort rule fields have to be reordered, interpreted and transformed to form a regular expression that could match an Internet packet structure. Because Snort rules are only applied to IP, TCP, ICMP and UDP Internet protocols an Internet packet treated for Snort will follow IP/TCP, IP/UDP and IP/ICMP protocol structure. A resulting regular expression for each kind of packet protocol will look like the regular expression frames shown in Fig. 1,2 and 3.

```
/^{Version+IHL{TOS{Total length}{Identification}
{Flags+Fragment}{TTL}{Protocol} {Header
Checksum}{Source Address}{Destination Address}
{Optional fields}{Source Port}{Destination Port}
{Sequence Number}.{Acknowledge Number}.{Data
Other + Reserved} {Flags}{Window}{Options}
{Payload}/
```

Fig. 1 IP/TCP regular expression frame

```
/^{Version+IHL}{TOS}{Total length}{Identification}
{Flags+Fragment}{TTL}{Protocol}{Header
Checksum}{Source Address}{Destination Address}
{Optional fields}{Source Port}{Destination Port}
{Options}{Payload}/
```

Fig. 2 IP/UDP regular expression frame

```
/^{Version+IHL}{TOS}{Total length}{Identification}
{Flags+Fragment}{TTL}{Protocol} {Header
Checksum}{Source Address}{Destination Address}
{Optional fields}{Type}{Code} {Checksum}
{Identification}{Sequence Number}{Options}
{Payload}/
```

Fig. 3 IP/ICMP regular expression frame

Each of these regular expression frames will be filled with the keywords found inside each Snort rule. Each snort rule has two parts, rule header and rule options. The rule header is composes of 7 fields: action, protocol, source address, source port, direction, destination address and destination port. The source and destination address could be implemented as a unique IP address or a range of IP addresses (CIDR, Classless Inter-Domain Routing, notation). The source and destination port could be implemented as unique or also a range of ports. The direction field will determine the source and destination addresses and port numbers in a rule.

The individual fields of the rule header part has to be interpreted and transformed into hexadecimal values to be included in the regular expression frame indicated in the protocol field. Snort2Regex include a configuration file that define Snort variables, for example many Snort rules refer to a variable $HOMENET as source or destination IP address, which must be defined in this configuration file.

Snort rule option fields are classified into 4 groups: general rule options, payload detection rule options, non-payload detection rule options and post-detection rule options.

General rule options provide information about the rule but do not have any effect during detection. They are composed of: msg, reference, gid, sid, rev, classtype, priority and metadata keywords. Snort2Regex generates an output file which can store all of the general rule option fields. This file can then be used by the target DPI system provide more information when an alert is triggered.

Payload detection rule options look are used when searching the packet payload and can be interrelated. These are composed of: content, nocase, rawbytes, depth, offset, distance, within, httpclientbody, httpcookie, httpheader, httpmethod, httpuri, fastpattern, uricontent, urilen, isdataat, pcre, bytetest, bytejump, ftpbounce, asn, cvs, dceiface, dceopnum and dcestubdata keywords.

Content refers to the data inside the packet payload. Its position inside the packet payload is determined by depth, offset, distance, within, httpclientbody, httpcookie, httpheader, httpmethod, httpuri, uricontent and isdataat. In this case a regular expression that tailors the content will include the "." meta character which indicates a space of 1 byte. Also the regular expression could be anchored if the option offset is found, as it indicates an absolute position inside the payload (See Fig. 4).

*(content:"ABC";offset:15;content:"DEF";distance:10)*

/^.{15}ABC.{10}DEF/

Fig. 4 Snort rule with offset keyword

Some rules could create several regular expressions, due to different content options that can't be tailored in a regular expression (See Fig. 5).

*(content:"ABC";content:"DEF";)*

*/ABC/*

*/DEF/*

Fig. 5 Snort rule which content can't be tailored

In that case the rule will trigger if both content are found in any position inside the payload. In that case we can't create a unique regular expression to emulate the behavior seen in rules like Fig.5. This solution would increase the number of rules but the system performance should not be affected as the resulting rule set would be less complex.

The PCRE keyword allows rules to be written using Perl Compatible Regular Expressions. This keyword is in most cases complementary to the content keyword and not a substitute. Another regular expression will be created if the content of the PCRE keyword can't be tailored with other content keywords found the Snort rule. The bytetest, bytejump, ftpbounce, asn, cvs, dceiface, dceopnum and dcestubdata keywords are still under investigation.

Non-payload detection rule options are used when searching non-payload data. Some Keywords are implemented in the regular expression frame as a class. This is necessary because Snort has the ability to check for a unique value or a range of values for each field. These keywords are fragoffset, ttl, tos, ipopts, fragbits, flags, window, itype, and icode. Fragbits and fragoffset share 2 bytes of the IP protocol header when both are in the same Snort rule, both has to be transformed and combined as indicated by the bit flags set and fragment offset values respectively.

Id, seq, ack, ipopts, flow, flowbits, icmpid and icmpseq keywords are transformed and integrated into the regular expression frame in their respective position, if they are found in the input Snort rule. The keyword ipproto indicates the protocol on the IP header. if this keyword is found in the input Snort rule, it is translated against a list of protocols and integrated into the output regular expression with the corresponding hexadecimal value. The Keyword sameip creates a rule that will match packets in which the source and destination IP addresses are the same. If this rule is found as the only option in the Snort rule, the output regular expression will be expensive to be processed as it requires backtraces in order to find the match. Flow, flowbits, Dsize, rpc and streamsize keywords are still under investigation.

Post-detection rule options are triggered after a rule has found a match. Composed by logto, session, resp, react, tag, activates, activatedby, count, replace, detectionfilter. Snort2Regex generates an output file which can store all of the post-detection rule options fields. This can then be used by the target DPI system provide more information when an alert is triggered.

## 5. TRANSFORMATION RESULTS

The Snort rule set version 2.8.5.3 used for the current implementation was released by SourceFire in 12 of August 2010. The result of the transformation generates 8701 regular expressions from the 3993 Snort rules used as source. To clarify the process of the transformation we will present three distinct examples. The first one shows how payload relative keywords content and PCRE are tailored to create the payload relative content of the regular expression frame. The second example shows how keywords relative to ICMP protocol header are interpreted. The last example illustrates the case of an IP/UDP relative rule that generates two regular expressions.

Example 1: The first example addresses a rule that detects a buffer overflow attempt in Oracle MySQL 5.0 through 5.0.91 and 5.1 before 5.1.47 allowing remote authenticated users to execute arbitrary code via a COM_FIELD_LIST command with a long table name (See Fig 6).

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3306
  (msg:"WEB-MISC Oracle MySQL Database
  COM_FIELD_LIST Buffer Overflow attempt";
  flow:to_server,established; content:"|04|"; depth:1; offset:4;
  pcre:"/[^\x0D\x0A\x00]{512}/Ri"; metadata:policy
  balanced-ips drop, policy security-ips drop;
  reference:cve,2010-1850; classtype:attempted-user;
  sid:16703; rev:1;)
```

Fig. 6 Snort rule sid 16703

This rule belongs to the IP/TCP regular expression frame. As part of the rule header we found source and destination IP addresses described with keywords that are interpreted with the information provided in the Snort configuration file. For this example we leave the "keywords" but users should provide the same information to the Snort2Regex configuration file to be translated. In this case destination port is also provided, if were indicated with a keyword the same approach as before will be used. This tool extracts the keyword information and fills the possible gaps inside the regular expression frame.

```
^.{12}$EXTERNAL_NET$HOME_NET(....){0-
10}.{2}\x{0CEA}.{12}
```

Fig. 7 Header content for Snort rule sid 16703

In the rule option we found a content keyword that will match the hexadecimal value "\x04" at the 4th byte of the payload. Also we found PCRE content with the modifiers "R" and "i", the "R" modifier is a special Snort dedicated modifier that has a similar functionality as "distance:0" and the "i" modifier has a similar functionality as "nocase" that will be inserted in the regular expression frame with \[\] wild cards . The tool interprets keywords, resolves any conflict between them and finally tailors them as is shown in fig. 8.

```
.{3}\x04\[^\x0D\x0A\x00]{512}\]
```

Fig. 8 Payload content for Snort rule sid 16703

To complete the regular expression we join both parts with a ".*" wildcard (See fig. 9), it is necessary to use IP/TCP and IP/ICMP regular expression frames as there are optional fields in the TCP and ICMP protocol that could makea separation of unknown length between the header and payload parts of the packets.

```
/^.{12}$EXTERNAL_NET$HOME_NET(....){0-10}.{2}
\x{0CEA}.{12}.*.{3}\x04\[^\x0D\x0A\x00]{512}\]/
```

Fig. 9 Regular expression for Snort rule sid 16703

Example 2: This rule is used as an example to detect the "SolarWinds ICMP and SNMP scanner" exploit tool as this tool advertises itself in the packet payload (See Fig 10).

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"SCAN SolarWinds IP scan attempt"; icode:0; itype:8;
content:"SolarWinds.Net"; classtype:network-scan; sid:1918;
rev:6;)
```

Fig. 10 Snort rule sid 1918

This rule belongs to the IP/ICMP regular expression frame. Source and destination IP addresses are described with keywords and source and destination port are described as any (ICMP protocol doesn't have port fields), we also found header relative keywords "icode" and "itype" that is translated and added to the regular expression frame. The optional field of the rule doesn't contain information about the content position inside the payload, the content is added to the regular expression frame and the ".*" wildcard is added (See Fig 11).

```
/^.{12}$EXTERNAL_NET$HOME_NET(....){0-10}
\x08\x00.{6}.*SolarWinds\.Net/
```

Fig. 11 Regular expression for Snort rule sid 1918

Example 3: The last example shows a rule that is used for detecting when a malformed communication attempt is sent to a VoIP device (See Fig 12).

```
alert udp $EXTERNAL_NET any -> $HOME_NET 5060
  (msg:"VOIP-SIP Call-ID header invalid characters
  detected"; content:"Call-ID|3A|"; nocase;  pcre:"/^Call-
  ID\x3A[^\r\n]+[\x01-\x08\x0B\x0C\x0E-\x1F\x80-
  \xFF]/smi"; reference:url,www.ee.oulu.fi/research/ouspg/
  protos/testing/c07/sip/;reference:url,www.ietf.org/rfc/rfc326
  1.txt; classtype:attempted-dos; sid:11993; rev:2;)
```

Fig. 12 Snort rule sid 11993

This rule belongs to the IP/UDP regular expression frame. Source and destination IP addresses are described with keywords and the destination port is provided. The optional field of the rule offers a "content" and also PCRE keywords. The rule doesn't have information about the position of the content keyword inside the payload, that means this content could be found in any position of the payload and it has to be separated from the header part of the rule with a ".*" wildcard (See Fig 13). We can see that the PCRE keyword starts with an anchor wildcard that clearly indicates the position of the PCRE content at the beginning of the packet payload (See Fig 14). As these two keywords can't be tailored together, this rule will generate two regular expressions with the same header part

```
/^.{12}$EXTERNAL_NET$HOME_NET(....){0-10}
.{2}\x{13C4}.*\[Call-ID\x3A\]/
```

Fig. 13 First regular expression for Snort rule sid 11993

```
/^.{12}$EXTERNAL_NET$HOME_NET(....){0-10}
.{2}\x{13C4}Call-ID\x3A[^\r\n]+[\x01-\x08\x0B\x0C\x0E-
\x1F\x80-\xFF]/smi
```

Fig. 14 Second regular expression for Snort rule sid 11993

In the last example we can see that the PCRE keyword extends the content keyword information looking for a more precise signature. Although for Snort performance it is important to use "content" as a filter. In this case the first regular expression isn't critical however in certain scenarios could be use as a filtering technique.

## 6. Conclusion and Future work

This paper addresses the problematic issue of multiple signature syntax used across multiple DPI implementations. A widely used syntax is suggested employing POSIX regular expressions, that can be exploited by hardware and software DPI implementations. To demonstrate the viability of this syntax this paper shows the novel tool "Snort2Regex" that benefits from the accuracy and versatility of the Snort rules collection. This work will allow general purpose regular expression matching engines for DPI to apply the Snort rule set by translating each rule into flexible regular expression syntax. This tool shows how Snort rule syntax can be transformed into POSIX regular expressions syntax without hindering the detection accuracy. Issues like rules redundancy are also explored. It is also discussed that features such as offset can avoid unnecessary computation.

For a future research redundancy and precision are the main issues in the short term, as well as other keywords implementation for Snort. Other signature syntax adaptation, coordination between different DPI systems and collaboration between rule writers are the long term issues that can be used to create a new context that allow users to develop security and application identification rules but lets them avoid the liability and requirements of a concrete DPI system. Intelligence should also be added to the system which could decide when to discard redundant rules or select them for filtering techniques. Also a labeling technique could be developed to indicate a clear separation between the header and payload of each packet in order to reduce unnecessary computation.

## I. References

[1] M. Roesch ''Snort–Lightweight Intrusion Detection for Networks,'' 13th Systems Administration Conference, USENIX, 1999.

[2] SourceFire , Inc., http://www.SourceFire .com/

[3] Cisco Visual Networking Index Global Forecast survey, 2010, http://www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html#~forecast

[4] Sommer, R. and V. Paxson, ''Enhancing Byte-Level Network Intrusion Detection Signatures with Context,'' ACM CCS, 2003.

[5] Paxson, V., ''Bro: A System for Detecting Network Intruders in Real-Time,'' USENIX Security, 1998.

[6] Bro Development Team, Bro official reference manual, International Computer Science Institute, http://www.bro-ids.org/wiki/

[7] Steven T. Eckmann. "Translating Snort rules to STATL scenarios". In Proc.Recent Advances in Intrusion Detection, October 2001.

[8] G. Vigna, S. Eckmann, and R. Kemmerer. The STAT Tool Suite. In Proceedings of DISCEX 2000, Hilton Head.,

[9] S. Eckmann, G. Vigna, and R. Kemmerer. STATL: An attack language for state-based intrusion detection. In Proceedings of WIDS(held in conjunction with ACMCCS2000), Athens, Greece, November 2000.

[10] K. Ilgun. USTAT: A Real-time Intrusion Detection System for UNIX. In Proceedings of the IEEE Symposium on Research on Security and Privacy, Oakland, CA, May 1993.

[11] G. Vignaand, R. Kemmerer. NetSTAT: A Network-based Intrusion Detection System. Journal of Computer Security, 7(1):37–71,1999

[12] A. Tongaonkar, S. Vasudevan, R. Sekar, "Fast packet classification for Snort by native compilation of rules", LISA'08.