A Multimedia Traffic Classification Scheme for Intrusion Detection Systems

Oge Marques and Pierre Baillargeon
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL – USA
{omarques, pbaillar}@fau.edu

Abstract

Intrusion Detection Systems (IDS) have become widely used tools for ensuring system and network security. Among many other challenges, contemporary IDS have to cope with increasingly higher bandwidths, which sometimes force them to let some data go by without being checked for possible malicious activity.

This paper presents a novel method to improve the performance of IDS based on multimedia traffic classification. In the proposed method, the IDS has additional knowledge about common multimedia file formats and uses this knowledge to perform a more detailed analysis of packets carrying that type of data. If the structure and selected contents of the data are compliant, the corresponding stream is tagged accordingly, and the IDS is spared from further work on that stream. Otherwise, an anomaly is detected and reported.

Our experiments using Snort confirm that this additional specialized knowledge results in substantial computational savings, without significant overhead for processing non-multimedia data.

1. Introduction

Intrusion Detection Systems (IDS) have become one of the tools of choice for ensuring system and network security. IDS usually scan ongoing traffic looking for patterns and/or signatures that might indicate malicious or unauthorized activity that should be investigated [1, 2].

One of the issues currently facing network-based IDS is the high computational cost of doing real-time analysis when a large amount of traffic is passing through a connection. In such cases, the IDS usually does not have any other option but to blindly skip certain packets [1].

This paper presents a novel method to improve the performance of IDS based on multimedia traffic classification. Under the proposed approach, the IDS has additional knowledge about common multimedia file formats and uses this knowledge to perform a more detailed analysis of packets carrying that type of data. If the data complies with the standard format, the corresponding stream is flagged and the remaining packets are ignored by the IDS, which can now focus on other traffic, therefore reducing the computational cost and making the IDS more efficient. Otherwise, an anomaly is detected and reported for further action by the system administrator.

This paper is organized as follows: Section 2 presents an overview of Intrusion Detection Systems, with emphasis on topics that are relevant to this work. Section 3 focuses on Snort, a popular open-source network intrusion detection tool used in our experiments. Section 4 explains the proposed method in detail. Section 5 shows results of experiments using Snort. Finally, we derive some conclusions and present directions for future work in Section 6.

2. Intrusion Detection Systems

"Intrusion Detection Systems are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems [3]". With the increase in the amount and severity of network-based attacks over the past few years, IDS have become an important and widely used additional tool in the network security infrastructure of many organizations. Several surveys and taxonomies for IDS have been published recently, such as [4]. In this section, we summarize some aspects of IDS that are directly related to this work.

2.1. Host-based vs. network-based

Intrusion Detection Systems can be fed data in a variety of manners. Multiple sensors spread across a network can report to a central IDS (network-based) or the IDS can be installed and monitor data on one device (host-based). These are generalized cases and there exist many configurations that combine both of these deployment techniques. The multimedia classification preprocessor described later in this paper works on any of these deployment schemes.

2.2. IDS strengths and limitations

An IDS enables a network administrator to deal with intrusions more efficiently by creating a centralized point where suspicious activity can be monitored and later analyzed with tools like ACID and Sguil. This eliminates the very time consuming process of manually checking log files and trying to identify suspicious activity.

While the use of IDS makes intrusion detection less of a chore, it must be noted that they cannot prevent intrusions. The usefulness of the IDS comes after an intrusion has taken place, allowing an administrator to retrace the steps of the intruder and find what security measures were bypassed.

3. Snort

Snort [5, 6] is a multi-OS, multi-platform network intrusion detection tool that has experienced great popularity during the past three years, mostly due to its extensible architecture and open source distribution. It was originally envisioned as a lightweight IDS, but it has evolved to become a full-featured, real-time IP traffic analysis and packet logging system.

Snort works by matching traffic patterns to its rules, stored in a *ruleset*. Once a packet enters through the Network Interface Card (NIC), it is decoded by a *packet decoder*, which determines which protocol is in use for a given packet and matches the data against allowable behavior for patterns of that protocol. If any anomaly (e.g., malformed headers, overly long packets, unusual or incorrect TCP options) is detected, an alert is generated.

After the packets are matched against the decoder, they are sent to (optional) *preprocessors*. Preprocessors are plug-ins for Snort that allow additional parsing and processing of incoming data. Snort users can write (in 'C') their own preprocessor modules, extending its functionality past the base ruleset to include features such as anomaly detection and session reconstruction. The ability to easily add functionality to Snort with preprocessors is offset by the computational overhead that is added with each preprocessor that Snort uses. Any preprocessor in Snort can be turned on or off simply by adding a line to a configuration file for the program.

The *detection engine* is the component of Snort that takes data from the packet decoder and preprocessors (if they are enabled) and compares it against the rules in the ruleset. After the rules have been matched against the data, Snort's logging mechanism allows archival of packets that triggered Snort rules, whereas its alerting mechanism is used to notify the system administrator that a rule has been fired.

Alerts in Snort can be displayed to the window the program is running in, logged to a text file, or entered into a database. Database logging is the most useful of these options, as it allows multiple Snort sensors to report alerts to one central location.

4. The proposed method

In this section we explain the proposed method in detail. We start by revisiting the research questions that prompted this work, and then proceed to explain it in more detail, with special attention to relevant implementation aspects.

4.1. Background

Work on this project began by attempting to answer the research question: "How can the performance of IDS – and, ultimately, the security goals that they attempt to achieve – be improved by incorporating knowledge of multimedia protocols, file formats, and headers into their operation?" More specifically: How much and what type of additional knowledge is needed? How should it be represented? What type of performance improvement can be achieved and how significant can it be? In the remainder of this section we hope to provide meaningful answers to these questions.

In order to consider how much and what type of additional knowledge is needed, let us start by looking at the current knowledge level upon which IDS decisions are usually made. Currently, IDS are capable of blocking multimedia content based on port number (streaming audio/video), string matching of content type (e.g., content: "User-Agent|3A| Quicktime") and file extension. None of these techniques verify the validity of the content; they simply assume that if data appears to be (from external identifiers, like MIME) multimedia, then it is.

This level of protection is basic, and we propose adding another level, which would use more header information to classify multimedia. This level (medium) could be the precursor to a higher level (expert) that would classify data based on a more in-depth knowledge of multimedia than simple header information and may contain some form of learning algorithm. The medium level of classification would entail examining the contents of an incoming data stream and looking for known multimedia characteristics, like JPEG or MPEG markers. Additionally, the IDS could analyze the markers to ensure they contained values within acceptable limits, checking to make sure, for instance, that the *frame rate* value is not a negative or excessively large number.

Once we had settled on adding this extra level of protection, our attention shifted to answering the next question: How should this knowledge be represented? Network-based IDS, such as Snort, have a customizable level of knowledge of the packets that are sniffed as they traverse the network. At the very basic level (implemented within the *packet decoder* module in Snort), the transport-level protocol is determined and the data is matched against allowable behavior for patterns under that protocol. Optionally, preprocessors perform a more detailed parsing and analysis of the data. Additional knowledge can be modeled by creating more preprocessor modules to handle those special cases. Therefore, the answer to the modeling and representation of additional knowledge had a simple answer within the Snort framework, which allowed us to implement a multimedia classifier as a Snort preprocessor (Figure 1).

Last, but certainly not least, we turned our focus to the expected performance improvement that could result from this additional level of knowledge and protection. Since we knew that the added functionality provided by preprocessors is offset by the associated computational overhead, we needed to devise a scheme in which we would make up for the additional CPU cycles needed to inspect multimedia traffic more closely. The answer to this challenge came in the form of a two-class classifier, whereas an incoming or outgoing stream is classified either as *multimedia* or *non-multimedia*.

Once a session is classified as multimedia traffic, Snort would be able to determine if an authorization was appropriate. By preprocessing and authorizing the data, we can take advantage of Snort's global **`do_detect`** flag that tells Snort to skip the detection phase (ruleset comparison) of the flagged packet. Since multimedia files are usually very large, the additional time spent classifying and flagging multimedia data using the first few packets would be offset by the time savings throughout the remaining packets corresponding to that stream, thereby achieving the intended performance improvements.

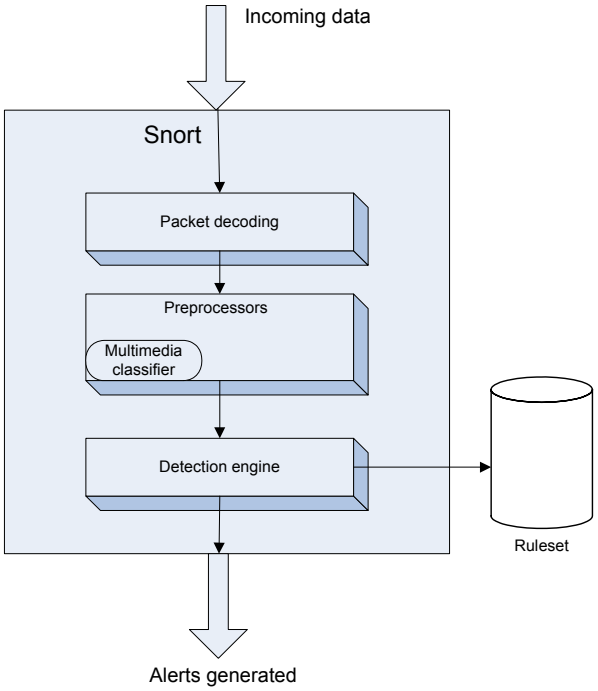

Figure 1 - IDS data processing

4.2. Goals and assumptions

Our method has the following primary goals:

(1) To save processor usage when dealing with (potentially large) legitimate multimedia files, allowing the IDS to focus on other types of traffic and minimizing the number of packets that may go by unchecked.

(2) To generate alerts when anomalous, multimedia-looking, traffic (e.g., a large file renamed with a **`.avi`** extension but not compatible with the AVI header rules) is detected.

The proposed scheme works under the following assumptions:

(1) We assume that multimedia traffic is naturally benign (which it appears to be currently – with the exception of the recent discovery of the JPEG exploit [7]).

(2) We assume that a network-based IDS, such as Snort, has been properly installed and configured to perform general intrusion detection tasks.

If assumption (1) turns out to be false in the future, our method should make it much easier to create IDS rules to properly deal with this traffic, providing, in a sense, a future proofing capability to the system.

4.3. How it works

Multimedia traffic classification is done with a Snort preprocessor, responsible for looking at traffic for multimedia identifiers, such as headers, to identify incoming multimedia traffic. The classification process (Figure 2) may be understood simply as a two-class classifier (*multimedia* and *non-multimedia*), but it can be extended to include a number of more specialized multimedia classifiers, depending on the users' needs (e.g., possibly for the purpose of detecting malicious activity in the future disguised as multimedia traffic).

Once a session is found to be containing multimedia traffic, the IDS is able to determine if an authorization is appropriate. Authorization will not necessarily be immediate, and may require multiple examinations of a data stream or session to determine content validity. If a stream is deemed to be unauthorized, normal IDS operation will continue and the data will be analyzed according to the remaining rulesets. The ultimate goal of the classification stage is a process that creates intelligent decisions based on previously encoded knowledge.

Possible outcomes of the multimedia classification stage are as follows:

(1) Data is recognized as multimedia stream and authorized, thereby allowing upcoming packets to bypass the IDS detection engine.

(2) Data is recognized as a *false* multimedia stream: an alert is generated to notify an administrator that a file has been transferred under possible false pretenses and the remaining packets in the stream bypass the IDS detection engine since they've already been determined to belong to a stream falsely posing as multimedia.

(3) The data is not multimedia in nature and continues through the IDS detection engine as it normally would if the multimedia classifier did not exist at all.

The first two scenarios result in significant processor savings, while the third has a slight increase in processor usage, since we have added a preprocessing stage before the detection engine which must analyze a few of the first packets of each new session.

After authorization, traffic containing multimedia content will be flagged and be able to bypass the IDS, thereby allowing the IDS to focus its resources analyzing the remaining traffic more thoroughly. This bypass mechanism is accomplished by recording the session number when a transmission is authorized, so consequent packets can be routed around the IDS (or through the IDS, but without the IDS having to do a comparison of the traffic against its ruleset).

Currently, Snort rules typically work by looking for traffic on well known multimedia ports, such as port 80, however someone could circumvent this easily by changing the server and clients to use a different port. The proposed multimedia classification scheme would make these types of rules much more accurate by flagging data based on content, not on packet headers.

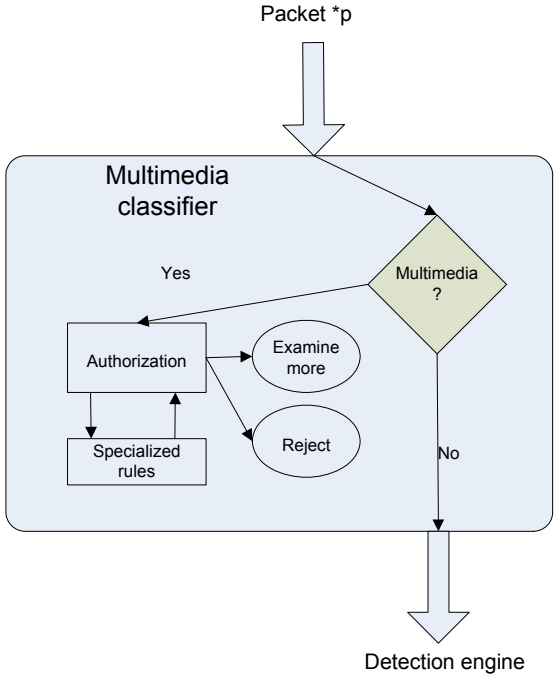

Figure 2 - Multimedia classifier

4.4. Implementation aspects

The idea behind our classification method is that multimedia traffic can be uniquely identified by certain characteristics, such as the header in an AVI file, and that once the traffic has been classified the remaining packets for the stream or session containing the multimedia content can bypass the analysis engine of the IDS. Our work relies on specialized knowledge of multimedia files. Currently, we have encoded limited knowledge of three popular file formats: JPEG, AVI, and MPEG.

JPEG headers consist of markers, which identify various parts of the header. These markers can be identified by one or more **`FF`** bytes followed by a marker byte. There are several marker codes of

interest, defining things such as the quantization table location and start of frame [8]. After a marker has been located, it is possible to examine the data that follows the marker to determine whether or not a file conforms to expected standards. For example, following the **FF C0** marker is the start of frame marker, which is followed by header length along with image height and width and number of components (3 for RGB, 1 for grayscale images).

AVI headers contain RIFF data chunks that can be parsed to retrieve information about an AVI movie. The most useful component for IDS classification purposes is the 'movi' header which contains number of frames, frame height/width, streams (audio/video), microseconds between frames and maximum data rate [9]. MPEG headers [10] are similar to JPEG in the structure using markers.

The following pseudocode explains the general relationship between Snort, the multimedia classification preprocessor and the detection engine (timing measurements for results in this paper were done using difference in execution time measured with the clock function):

```
snort_main() {
   capture_packets();
   extract_packet_data();
   for_all_packets {
      preprocessors(packet *p);
     detection_engine(packet *p);    } }

   /* Run packet through detection engine */
   detection_engine(packet *p) {
   start=clock();
   if(do_detect)
     check_packet_against_ruleset(packet *p);
   stop=clock();
   det_cpu_usage+=stop-start; }

   /* Run preprocessors */
   preprocessors(packet *p) {
     frag2(packet *p);
     stream4(packet *p);
     etc(packet *p);
     multimedia_classifier(packet *p); }

   /* Analyze packet to see if it contains
   multimedia data */
   multimedia_classifier(packet *p){
     start_mm=clock();
     while(!at_end_of_packet_data) {
        if(current_byte ==
        known_multimedia_marker) {
            if(next_byte ==
            known_mm_marker_part_2) {
              /* More checks until header
              sequence is verified */
                do_detect=0;
               // Stream is authorized
                 }  }   }
     stop_mm=clock();
     mm_cpu_usage+=stop_mm-start_mm; }
```

5. Experiments and results

Our approach was tested by monitoring FTP transfers of multimedia and non-multimedia files using Snort. All experiments had the same common goal: to measure processor usage and evaluate the impact of adding the proposed specialized preprocessor on the overall performance of the IDS. The pseudocode indicates which submodules are involved and where exactly the time measurements are done.

5.1. Single file transfers

In the first series of experiments we looked at specific file types, one at a time. For AVI files varying from 716 KB to 56 MB in size, a fixed number of packets (two) per file was required to classify them as valid multimedia data. The corresponding savings in processor usage was somewhat inversely proportional to the file size: for small files, the processor was used only 15% of the time it would have without our classifier, while for very large files, this number would drop to less than 1%. Similar results were obtained for MPEG and JPEG files between 417 KB and 67 MB in size. These experiments also confirm that the overhead introduced by adding an extra preprocessing step is minimal – less than 1%.

5.2. Mixed traffic

These experiments combine multimedia and non-multimedia files into a batch FTP job. Multiple files were transferred with the multimedia classification preprocessor active and then again without to evaluate CPU utilization. Figure 3 summarizes the results, indicating that processor usage is inversely proportional to the amount of multimedia data in the batch.

6. Conclusions and future work

We have proposed and implemented a multimedia traffic classification scheme that provides additional knowledge to, and extends the functionality of, a network-based IDS (such as Snort). It was implemented as an extra preprocessor module for Snort and tested with several combinations of multimedia and non-multimedia traffic.

Currently, IDS systems, such as Snort, usually are able to block files based on file extension. With the proposed scheme, it is possible to block based on the actual content of the files.

The proposed method helps system administrators in two main ways:

(i) for legitimate multimedia traffic, it tags the corresponding stream and saves processor usage by not looking at subsequent packets of the same stream;

(ii) for anomalous traffic (e.g., a large file renamed with an AVI extension but incompatible with the AVI header rules), it generates the corresponding alerts.

Administrators will have more accurate reports, and hopefully a higher detection rate of intrusions because the IDS will be inspecting more relevant information (non-multimedia), when resources become saturated. Specific examples include encrypted (as in the case with SSL connections) and out-of-sequence packets. Without any authorization scheme an IDS must decrypt or re-order the incoming data for analysis, whereas authorized traffic would be allowed through without needing this extra effort.

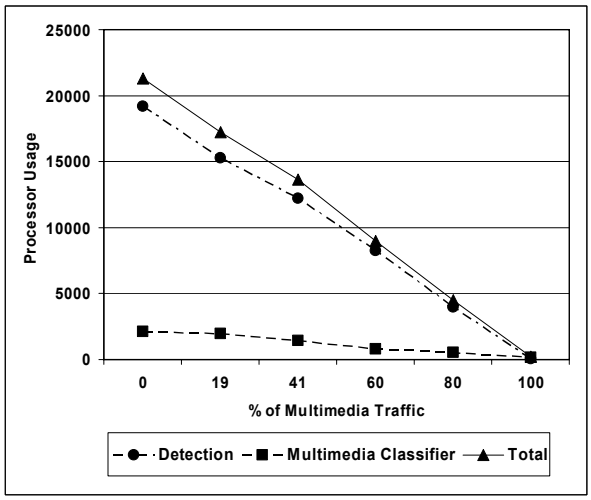

Figure 3 – Processor usage versus percentage of multimedia traffic

Results of our experiments confirm that significant processing savings can be gained, given that there is minimally a small amount of multimedia traffic on a network. The results indicate that this is a promising avenue worth pursuing a bit further.

The current work has a number of limitations which should be relaxed over time, particularly:

(i) It assumes file downloading / uploading via FTP or HTTP. Extension to streaming is being investigated and implemented.

(ii) Current file format knowledge is limited to JPEG, MPEG, and AVI. Extension to other popular file formats (e.g., WMV and MOV) is under way.

(iii) The amount of knowledge encoded into the classification stage is currently limited to static header markers. Adding an extra level of knowledge to enable analysis of header parameters such as valid frame rate, width, and height, etc., will be implemented soon.

Future work may also include the design and implementation of a machine learning scheme by which rules are created and updated based on observed traffic.

Acknowledgment

This work was partially supported by a grant from the U.S. Department of Defense (DoD).

References

[1] R. Bace, "An Introduction to Intrusion Detection & Assessment" (White paper), *ICSA Labs*, January 2000.

[2] R. A. Kemmerer and G. Vigna, "Intrusion Detection: A Brief History and Overview", *IEEE Computer*, Vol. 35, Issue 4, April 2002, pp. 27-30.

[3] R. Bace and P. Mell, "Intrusion Detection Systems" (Technical Report), *National Institute of Standards and Technology*, November 2001.

[4] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy" (Technical Report 99-15), *Department of Computer Engineering, Charlmers University*, March 2000.

[5] J. Beale et al., Snort *2.1 – Intrusion Detection (2nd ed.)*, Syngress Publishing, Rockland, MA, 2004.

[6] Snort web site: http://www.snort.org/

[7] "New, dangerous Microsoft JPEG exploit released". http://www.infoworld.com/article/04/09/23/HNnewjpegexploit_1.html

[8] JPEG Header information: http://www.obrador.com/essentialjpeg/headerinfo.htm

[9] AVI Header information: http://pvdtools.sourceforge.net/aviformat.txt

[10] J. Mitchell et al., *MPEG Video: Compression Standard (Digital Multimedia Standards Series)*. Kluwer Academic Publishers, October 1996.