AN INTELLIGENT MULTIMEDIA TRAFFIC CLASSIFIER WITH SECURITY APPLICATIONS

Oge Marques, Alvaro Fonseca, and Liam M. Mayron

Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL - 33431 - USA

ABSTRACT

The volume of multimedia traffic transmitted over the Internet has increased significantly over the past few years. As a result, the need to analyze and classify multimedia traffic has become more acute. Conventional packet- or header-based analysis has limitations; a deeper understanding of the actual multimedia content may be necessary. This paper presents a machine learning-based multimedia traffic classification solution that performs content analysis on video sequences with potential applications in security. Experimental results show that the proposed solution can classify video sequences as suspicious based on inspection of their visual content.

1. INTRODUCTION

The increase in multimedia traffic over communication networks has introduced a number of new challenges. Multimedia traffic accounts for a significant percentage of total Internet traffic, both in terms of streaming and downloading [1, 2]. Network administrators need to monitor multimedia traffic for security purposes and are not always equipped to do so effectively. Certain applications, particularly video streaming, require that the visual quality of the contents be monitored as they traverse the network. For these, and several other cases, a technical solution by which multimedia traffic can be classified in an automatic way (e.g., *suspicious* vs. *benign*) may be beneficial.

This paper presents a new method for implementing multimedia traffic classification with content analysis capabilities. It expands upon the ideas described in [3]. It is demonstrated in the context of video transfers and can be extended to support additional formats, metrics, protocols, standards and machine learning algorithms.

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

2. BACKGROUND

2.1. Motivation

The ideas proposed in this paper have been motivated by the following:

1. The usage of multimedia contents has grown significantly in recent years and will continue to increase as broadband Internet access and mobile computing technologies become mainstream. Multimedia traffic consumes a significant amount of resources on the Internet. Since image, audio, and, especially, video files are relatively large (even in their usual compressed form) multimedia traffic accounts for a significant percentage of the total Internet traffic, both in terms of streaming and downloading [1, 4, 2, 5, 6].

2. Even though multimedia traffic has usually been considered benign, the number of multimedia-related security exploits has grown significantly over the past few years [7, 8, 9, 10, 11, 12, 13]. As multimedia formats and algorithms become more complex there is a greater chance that their implementation may introduce new security vulnerabilities. There have been recent examples of worms being embedded in video streams [14] as well as an increasing need to detect video content that has been tampered with [15].

3. Multimedia traffic analysis is also needed in applications unrelated to security, particulary video quality monitoring in the presence of network impairments.

4. The tremendous popularity of relatively new services such as Google Video [16] and YouTube [17] indicates that video is becoming a mainstay of network traffic.

2.2. Related work

There have been a number of studies reporting on multimedia traffic, both streaming and downloading, and their characteristics. These studies provide insight on which types of media and traffic patterns merit further investigation.

Mena and Heidemann performed a study of Real Audio traffic over the Internet [6]. They note that multimedia traffic is indeed different from traditional Internet traffic. While the bit rate of multimedia traffic is consistent over a time span of minutes is far less predictable at smaller time scales. However, these sessions are generally lengthy (for example, an audio session has an average duration of 78 minutes) [6]. Multimedia transfer sessions are overwhelmingly unidirectional, with the data ratio from the server to the client being as high as 50 to 1 [6]. They conclude that the duration of audio sessions as well as the consistent packet length and interdeparture regularity are key characteristics that can be used to identify such multimedia flows.

Li et al. analyzed network traffic to determine the properties of Internet-bound streaming media. They concluded that codecs associated with RealNetworks were the most popular, followed by, in order, MP3, Apple, and Microsoft codecs [4]. According to their research the majority of streaming media is videos and over 90% of these videos are formatted specifically for the higher bandwidth of broadband connections [4]. The bandwidth required for streaming video is set to continue increasing as video resolutions grow.

Wang et al. analyzed the performance of RealVideo over the Internet in 2001 [5]. They concluded that performance is limited by the bandwidth capacity of the end-user but that the bottleneck is slowly shifting to the bandwidth capacity of the server.

Van der Merve et al. performed an empirical study to characterize streaming video traffic over a network [18]. Their work made several relevant observations. In terms of streaming video, high bandwidth media is requested more than twice as often as the corresponding low bandwidth version. High bandwidth media consists of 94% of streaming traffic. When given the choice between Windows Media and Real-format media 75% of requests choose Windows Media. Seventy percent of the transferred information is sent over TCP. The remainder is sent over UDP. Interestingly, 6000 hours (as of the time their work was published) of streaming media is created on a weekly basis [18].

Marques and Baillargeon [3] have proposed a multimedia traffic classifier that works in connection with an Intrusion Detection System (IDS). Their work addresses one of the issues currently facing network-based IDS, namely the high computational cost of doing real-time analysis when a large amount of traffic is passing through a connection. In such cases IDS usually have no choice but to skip packets. They have demonstrated that adding specialized multimedia knowledge to an IDS' packet analysis capabilities may help detect multimedia specific exploits (and, as a bonus, achieve substantial computational savings) in both streaming and non-streaming scenarios. They have presented a method to improve the performance of IDS based on multimedia traffic classification: by embedding multimedia-specific knowledge into the IDS, trusted multimedia contents can be identified and allowed to bypass the detection engine, thereby allowing the IDS to focus on other traffic. Moreover, IDS become capable of detecting multimedia-specific exploits which would otherwise go by unnoticed. The proposed method improves the work on multimedia traffic classification described in [3] in two ways:

1. Replacing hard-coded rules with the functionality and versatility of machine learning.

2. Moving beyond header-based analysis into pixel-based measurements.

The work in [3] was limited in that it required a snapshot of expert knowledge to be captured and implemented. With the constantly-changing nature of security threats this is not always a practical solution. At the very least, this limitation merited the investigation of a more flexible machine learning approach. Additionally, it did not perform analysis beyond the media's header information. With new security vulnerabilities that leave the header unaffected but alter the underlying content it was necessary to develop a new method that has the ability to inspect the content of the multimedia data in transit.

2.3. Technical challenges behind multimedia traffic classification

The decision of how to best represent multimedia knowledge can be approached from two directions: *breadth* (file formats, protocols, and their variants), and *depth* (the protocol/format structure).

2.3.1. Breadth

The vast amount of file formats and protocols used for multimedia communications introduces a problem that is compounded by the reality that many of these are proprietary and that different codecs may implement specific variations of the reference document for that format or protocol. The proposed approach has been implemented to work with some of the most popular image, audio, and video formats – namely JPEG, GIF, PNG, WAV, MP3, MPEG, AVI, and MOV – and the RTSP streaming protocol, but its framework can be easily extended to a wider range of supported formats and protocols.

2.3.2. Depth

Once a certain format or protocol is implemented the issue becomes one of considering depth, specifically, how thoroughly should one inspect headers, contents or layers before it can make an informed decision. Here we are faced with a design decision that parallels the one reported in [19]: deciding the different depths at which we can examine video packets to extract relevant information. In [19] this is done for the purpose of video quality analysis, whereas in our work the

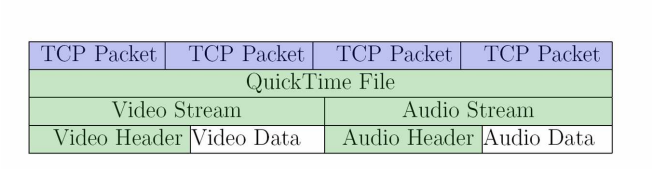

Fig. 1. Packetization of a video stream in multiple layers (an example using QuickTime).

goals are more general and may include security-driven applications (e.g., improved intrusion detection capabilities and firewall operation).

The different depths can be mapped to different layers in the protocol stack. Figure 1 shows a typical encapsulation of the various protocol layers in a packetized video stream. This example uses Apple QuickTime, but the basic principles are general enough to be applied to other protocol stacks, e.g., Windows Media. Starting from the top-most layer, the deeper we inspect the contents of a packet, the more detailed information we can obtain about the corresponding video stream and the more accurate inspection of possible intrusion-related abnormalities we can perform.

Inspection of the TCP/IP packets, without any further analysis, is the baseline case. On the opposite end, there would be a full analysis of the associated video stream after having been fully decoded.

3. PROPOSED MODEL

3.1. Design

The proposed model is a multimedia classifier whose input is a stream of packets transmitted over a network and whose output is a classification of those packets. The multimedia classifier relies on a set of specialized rules that encode the knowledge that is needed to make such a classification decision. The ultimate goal of the classification stage is to be able to make intelligent decisions based on previously acquired knowledge that is dependent on the actual video content.

In the proposed model, once a transfer of multimedia data is identified, its packets are inspected and bitstream information is obtained. A *non-reference method* is required, because in most cases the original video is unknown. Reibman and colleagues [20], [21] propose the use of blockiness, blurriness and ringing artifacts for assessing the quality of individual images that comprise the video in a non-reference framework. *Blurriness* is a measure of high-frequency content that may have been attenuated due to recording aspects of the transmitted video as well as compression methods [20].

In this paper, we extend the use of metrics such as blurriness, from a quality-oriented scenario to a security-driven one. The rationale behind using a metric such as blurriness for security purposes stems from the fact that it is possible that a seemingly innocuous video stream may actually contain malicious non-visual information. While in the previous work [3] we have taken preventive measures to detect such malicious traffic at the header level, in the present solution we extend it to the actual video contents. In this particular paper, we have chosen to use blurriness as an example of measure to detect "video whose quality is unacceptable, and therefore, suspicious".

There is another – more subtle – scenario, in which a few frames within the sequence are of significantly lower quality than the others. This "selective blurring" of isolated frames (with a possibly malicious intention) will easily fool the human eye, due to its inability to resolve detail within the frame duration.

We have performed empirical experiments that show that – independently of video quality – frames in the same scene or even in the same movie have similar blurring measures. It is, therefore, suspicions that a single frame presents a different measure of blurriness: it seems as if the frame is not part of the video. To address this possibility, we have calculated the selected metric (blurriness) for the entire sequence. The computed average blurriness and the standard deviation are then used to classify the video sequence as either *benign* or *suspicious*. Suspicious data can be flagged for more detailed processing, but that is beyond the scope of this work.

3.2. Implementation

The current implementation is bound by the following:

1. Media formats: we limited our work so far to AVI video files. However, because the analysis is performed in the uncompressed domain other video formats can be used.

2. Metrics: we limit this work to two metrics – average blurriness of a video sequence and the standard deviation of blurriness across all frames in a particular video sequence. Since it is our objective to show the utility of inspecting packets at the application layer, we chose metrics that cannot be determined otherwise.

3. Machine learning: the requirements of this experiment (a two-class classifier – benign and suspicious video) were satisfied with a J48 decision tree. We used Wekas J4.8 algorithm. It is a Java implementation of Quinlans classic C4.5 decision tree learner [22].

4. Applications: while a principal application of the proposed model is security, it can be used for other purposes, such as quality analysis.

The combination of tools provided by the NetAI package [23] motivated the architecture of this work. NetAI bridges the gap between a packet capture tool (NetMate) and machine learning analysis (using Weka). Our implementation is based on this architecture, however, it replaces NetMate and NetAI with other tools such as:

- Libpcap for packet capture
- Mplayer for media formatting
- MATLAB (on a separate platform) for data analysis
- Weka for machine learning

Libpcap [24] offers the possibility for accessing session, data format and application layer information. Any filtering requirement can be accomplished by using Libpcap library instead of the rigid shell commands offered by NetMate. It provides a low-level access to the data that passes through the selected network interface. The desired frame within the bitstream is obtained through a Linux interprocess communication procedure from Libpcap to Mplayer. Afterwards, raw data is analyzed and classified by performing MATLAB image processing algorithms and Weka classification methods.

Two generally parallel processes occur in our model. They are illustrated in Figure 2. Initially, machine learning algorithms have to be trained in order to achieve a proper classification, which means that a data set file must be created according to Weka file format specifications. The ground truth on the training model is a subjective judgment of a suspicious or benign video based on its perceived quality (degree of blurriness and infected frames). The training data set is transmitted, captured and filtered using a Libpcap-based sniffer. Data format and application contents are sent to Mplayer for data formatting, which is sent as an input to MATLAB routines for blurriness measurement. The resulting dataset file (ARFF file for machine learning language) is then formatted and loaded into the classification algorithms.

After the system has been trained, it can take unknown media information and classify it as either benign or suspicious. Sniffing, filtering and blurriness measurements are accomplished in the same way that was used to obtain the training data set.

4. CASE STUDY

4.1. Methodology

Our dataset consisted of 210 different video clips based on seven video test sequences. For each sequence six different levels of blurriness (motion blur) were generated (none, 5, 10, 15, 20, and 25). For each sequence and at each different degree of blurriness five different amount of frames were removed (0, 2, 4, 6, 8) and randomly replaced with frames from other sequences. All sequences were cropped to 100 frames.

For all sequences the overall blurriness and the standard deviation of the blurriness across all frames was calculated. All sequences were manually classified as *viewable or not viewable* (either for training or verification). The training dataset comprised 20% of the overall data (36 sequences), chosen at random.

Fig. 3. The original video frame - the blurriness is 2.8330.

We used Weka with the training dataset to generate a J48 decision tree model for analysis. We then gathered true positives, true negatives, false positives, false negatives, precision and recall on the remainder of the data.

This implementation is intended to measure blurriness in non-streaming video media. One application of this metric is for the classification of media as normal (benign) or suspicious, depending on the calculated degree of blurriness. This is insight that cannot be gained from the traditional inspection of only header information.

In this experiment data was transferred using the File Transfer Protocol (FTP). Since the specifications of FTP are known, we were able to easily strip the data to access and record the payload when part of a file is being transferred. Sampling terminated once the FIN flag is encountered.

The acquired binary information must be properly reformatted prior to analysis. Linux shell commands, and MPlayer [25] were used to ensure the file was properly formatted as an AVI file.

The key step, measuring blurriness, was performed offline in Matlab. The chosen metric, proposed by [26] first detects vertical edges in grayscale frames and calculates blurriness as the distance between the relative maximum and minimum points for each edge. The mean of all calculated points is the blurriness. The total video blurriness is the arithmetic mean of the individual blurriness for all video frames. The standard deviation was calculated with `std.m` function in Matlab using all blurring measurements for a whole video. Images are blurred with `fspecial('motion',LEN,THETA)` in Matlab where `LEN` determines the amount of motion (blurring) and `THETA` direction of the motion. Figures 3 and 4 show two versions of the same frame blurred differently.

The Weka data mining tool was used for learning. We created the requisite ARFF source files for the training dataset, with the single attribute of blurriness used to distinguish benign from suspicious video in one test case. In a second test case a measure of standard deviation is also used to generate

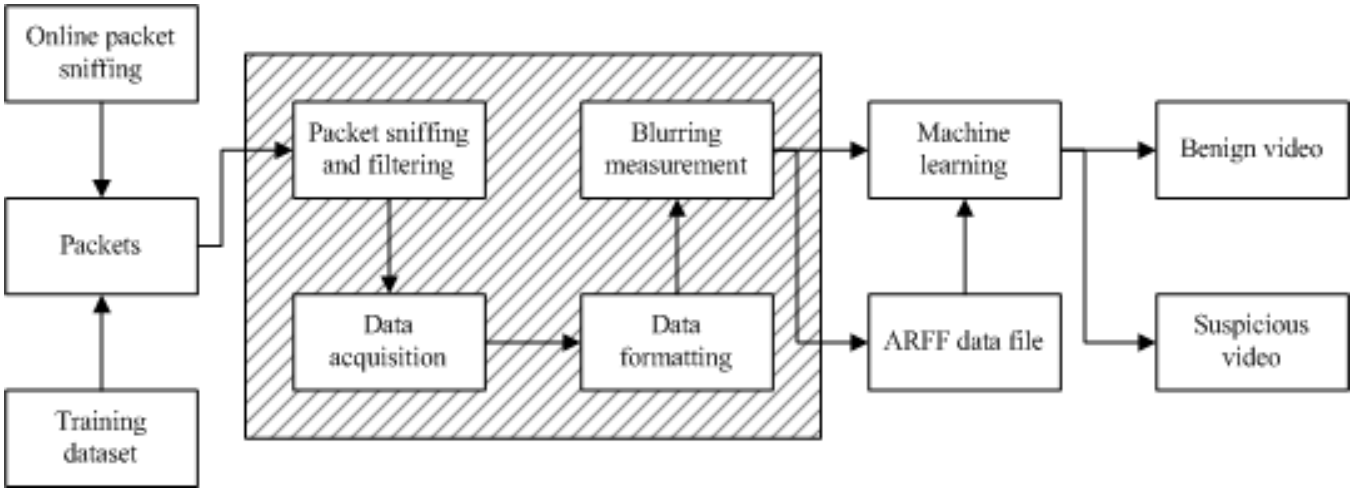

Fig. 2. System block diagram.

Fig. 4. The video frame after blurring - the blurriness is 21.1812.

the model. A J48 decision tree was the model used.

4.2. Results

Table 1 shows results for classification when only the blurriness measure is considered.

Table 1. Results for the blurriness-only model

	TP Rate	FP Rate	Precision	Recall
Benign	1	0.317	0.333	1
Suspicious	0.682	0	1	0.683

Table 2 shows results for classification when the model is generated based on only on the overall sequence's blurriness, but on the standard deviation of the blurriness of individual frames as well.

Our experiments show that, in this case, the results are significantly improved with the additional metric (standard deviation of blurriness values). As stated earlier, frames in a video sequence usually have similar blurriness; the blurriness standard deviation metric accounts for the possibility that some frames may have been replaced, improving the baseline case (average blurriness only).

Table 2. Results for the model derived from both average blurriness and the standard deviation of blurriness across all frames

	TP Rate	FP Rate	Precision	Recall
Benign	1	0.104	0.659	1
Suspicious	0.896	0	1	0.896

5. CONCLUSIONS AND FUTURE WORK

We have proposed a new solution for multimedia traffic classification in which the information at the application layer is used to provide additional insight beyond what is possible to extract from the header level. We selected blurriness as a demonstrative metric that can be used as an indication of suspicious video data, but our method is general enough to be extended to a variety of metrics. The work described in this paper can be expanded upon to provide a more flexible platform for multimedia traffic classification along the following directions:

1. A larger dataset
2. A realtime implementation of the blurriness metric
3. Additional metrics
4. Additional video formats
5. Integration with an IDS-based solution.

6. REFERENCES

[1] L. Guo, S. Chen, Z. Xiao, and X. Zhang, "Analysis of Multimedia Workloads with Implications for Internet Streaming," in *WWW2005*, Chiba, Japan, May 2005.

[2] J. van der Merwe, S. Sen, and C. Kalmanek, "Streaming Video Traffic: Characterization and Network Impact," in *Proceedings of the Seventh International Web Content Caching and Distribution Workshop*, Boulder, CO, August 2002.

[3] O. Marques and P. Baillargeon, "A Multimedia Traffic Classification Scheme for Intrusion Detection Systems," in *Proc. of the IEEE ICITA'05*, Sydney, Australia, July 2005.

[4] M. Li, M. Claypool, R. Kinicki, and J. Nichols, "Characteristics of Streaming Audio and Video Stored on the Internet," Department of Computer Science, Worcester Polytechnic Institute, Worcester, Massachusetts, Tech. Rep. WPI-CS-TR-03-18, May 2003.

[5] Y. Wang, M. Claypool, and Z. Zuo, "An empirical study of RealVideo performance across the Internet," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, San Francisco, California, USA, November 2001.

[6] A. Mena and J. Heidemann, "An empirical study of real audio traffic," in *Proceedings of the IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.

[7] "Apple QuickTime FlashPix integer overflow," http://www.kb.cert.org/vuls/id/570689.

[8] "Apple QuickTime JPEG integer overflow," http://www.kb.cert.org/vuls/id/289705.

[9] "Apple QuickTime MPEG-4 movie buffer overflow," http://www.kb.cert.org/vuls/id/587937.

[10] CoreLabs, "Core Security Technologies Advisory: MSN Messenger PNG Image Parsing Vulnerability," http://www.coresecurity.com/, 2005.

[11] Microsoft, "Microsoft Security Bulletin MS04-028: Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987)," http://www.microsoft.com/technet/security/bulletin/MS04-028.mspx, 2004.

[12] ——, "Microsoft Security Bulletin MS05-009: Vulnerability in PNG Processing Could Allow Remote Code Execution (890261)," http://www.microsoft.com/technet/security/bulletin/MS05-009.mspx, 2005.

[13] Mozilla Foundation, "Mozilla Foundation Security Advisory 2005-30: GIF heap overflow parsing Netscape Extension 2," http://www.mozilla.org/security/announce/mfsa2005-30.html, 2005.

[14] R. Lemos, "Malware goes to the movies," http://www.channelregister.com/2006/11/16/movies_gets_malware/.

[15] K. Greene, "Detecting video forgeries," http://www.technologyreview.com/InfoTech/17835/page1/.

[16] "Google Video," http://video.google.com.

[17] "YouTube - Broadcast Yourself," http://youtube.com.

[18] J. Van der Merwe, R. Cceres, Y. Chu, and C. Sreenan, "Mmdump - a tool for monitoring Internet multimedia traffic," in *ACM Computer Communication Review 30(4)*, Boulder, CO, October 2000.

[19] A. Reibman, S. Sen, and J. van der Merwe, "Network Monitoring of Video Quality over IP," in *Picture Coding Symposium, Special Session on Video over Networks*, San Francisco, CA, December 2004.

[20] A. Leontaris and A. Reibman, "Comparison of blocking and blurring metrics for video compression," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005 (ICASSP '05)*, vol. 2, Philadelphia, PA, March 2005, pp. 585– 588.

[21] A. R. Reibman, V. A. Vaishampayan, and Y. Sermadevi, "Quality monitoring of video over a packet network," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 327–334, 2004.

[22] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[23] "netAI," http://caia.swin.edu.au/urp/dstc/netai/.

[24] "libpcap," http://www.gsp.com/cgi-bin/man.cgi?section=3\&topic=pcap.

[25] "mplayer," http://mplayerhq.hu/.

[26] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi, "A no-reference perceptual blur metric." in *ICIP (3)*, 2002, pp. 57–60.