# Dialog-based Payload Aggregation for Intrusion Detection

Tobias Limmer and Falko Dressler
Computer Networks and Communication Systems
Department of Computer Science, University of Erlangen, Germany
{limmer,dressler}@cs.fau.de

## ABSTRACT

Network-based Intrusion Detection Systems (IDSs) such as Snort or Bro that have to analyze the packet payload for all the received data show severe performance problems if used in high-speed networks. Recent research results improve pattern matchers based on efficient algorithms or using specialized hardware. We approach the problem in a completely different way by considerably reducing the amount of data to be analyzed with only marginal impact on the detection quality. Dialog-based Payload Aggregation (DPA) uses TCP sequence numbers to decide which parts of the payload need to be analyzed by the IDS. Whenever a connection starts, or if the direction of the data transmission between peers changes, we forward the next $N$ bytes of traffic to an attached IDS. All data transferred after the window is discarded. Our analysis using live network traffic and multiple Snort rulesets shows that most of the pattern matches occur at the beginning of connections or directly after direction changes in the data streams. According to our experimental results, our method reduces the data rate to be processed to around 1 % in a typical network while retaining more than 98 % of all detected events. Assuming a linear relationship between the data rate and processing time of an IDS, this results in a speedup of two magnitudes in the best case.

## Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*;
C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Monitoring*

## General Terms

Measurement, Security

## Keywords

Aggregation, Intrusion Detection, Monitoring

## 1. INTRODUCTION

Network attack detection using signature matching on payload, also called Deep Packet Inspection (DPI), produces very good results in detecting current malware and other network attacks, although a slowly increasing number of malware types uses encrypted communication (e.g. the Storm malware). Intrusion Detection Systems (IDSs) using DPI like Snort or Bro struggle to keep up with processing all monitored network packets, because payload signature matching induces high processing requirements. This problem is especially difficult in networks beyond 1 Gbit/s [1]. Many rules for IDS already use network and transport layer header data to restrict the amount of traffic to be processed. Other solutions involve utilizing specialized hardware [4], parallelization of the analysis, and improved matching algorithms.

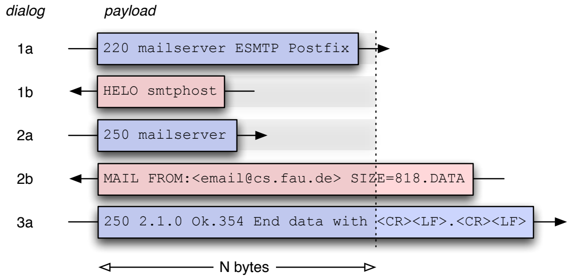
dialog
payload
1a
220 mailserver ESMTP Postfix
1b
HELO smtphost
2a
250 mailserver
2b
MAIL FROM:<email@cs.fau.de> SIZE=818.DATA
3a
250 2.1.0 Ok.354 End data with <CR><LF>.<CR><LF>
N bytes

**Figure 1: Dialog-based payload aggregation for TCP streams**

If header information is used for selecting individual connections for further analysis, systems like Snort still have to parse the complete payload of the entire connection, no matter whether the data is relevant or not. We argue, that most of the security-relevant data is transmitted at the beginning of a connection, in contrast to the following bulk data. One of the first approaches that stripped network connections and only stored the first $N$ byte of each direction was proposed in [2]. We used a similar idea, called Front Payload Aggregation (FPA), in [3] and extended it for filtering network data for intrusion detection. Nevertheless, we also realized that IDS frequently fails for protocols like HTTP or SMTP, which are basically used to transport higher-layer application protocols.

In this work, we describe the key reasons for this problem and introduce a completely new technique named Dialog-based Payload Aggregation (DPA). Contrary to the previously described approaches, DPA selects packet payload not only from the beginning of a connection, but also from later stages of a connection. In particular, DPA identifies direction changes within the bidirectional data stream using TCP sequence numbers. According to our experimental evaluation, DPA shows much higher detection ratios when combined with network-based IDS compared to the mentioned approaches.

*CCS'10,* October 4–8, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

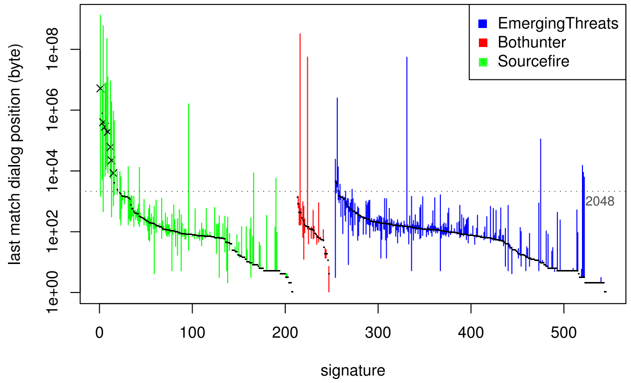

**Figure 2: End position of matches relative to dialog change in TCP connections**

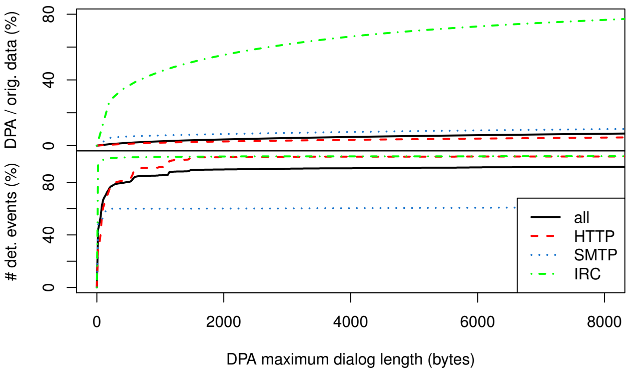

**Figure 3: Achieved data reduction rates and corresponding IDS detection rates by DPA**

Additionally, the length of dialog segments as identified by DPA can be used a new measure for identifying application protocols or attacks, as well as for statistical anomaly detection.

## 2. METHODOLOGY

FPA [3] offers a lightweight algorithm that aggregates a connection's first $N$ bytes of both directions based on the sequence number for TCP streams, or the order of packets in UDP streams. Most of the security-relevant data can be retained this way, but for protocols that exchange control and bulk data within the same connection in an interleaved way, important data may be lost during the aggregation process. A typical example is HTTP, which supports pipelining, where multiple requests may be sent within the same TCP connection. Here, FPA usually only captures the first request and response.

DPA improves the data filtering stage by using transport layer information to identify dialog elements within a TCP connection. When data is sent in a TCP stream, the sequence counter of the data sender is increased. By watching the sequence counters in both directions, we can identify the point in time when the transfer direction changes. Exploiting this knowledge, DPA captures the first $N$ bytes after each direction change. The principle is shown in Figure 1. DPA allows to capture contents of each dialog segment and to obtain much better results with protocols that mix control and bulk data.

## 3. EVALUATION

We performed a preliminary analysis of DPA using a modified version of Snort: We captured ten timeslices per day of 10 minutes of our university's Internet uplink for a period of 2 months and processed the data using a modified version of the IDS Snort. We selected three different rulesets for this analysis: the dataset shipped by the creator of Snort, Sourcefire, the open-source ruleset EmergingThreats and the ruleset used by the malware detection system Bothunter. Of these rulesets, we only included rules matching TCP payload data in our analysis. We recorded the end position of each match relative to the last direction change of the dialog.

Figure 2 shows the results: For each plotted signature ID, we recorded at least 10 matches. The match end position is shown on the logarithmic-scale y-axis, and the corresponding signature ID is shown on the x-axis. We performed the test for 3 different rulesets coming from Sourcefire, the open-source initiative EmergingThreats, and the malware detection system Bothunter. A black dot marks the median and the vertical line marks the upper and lower 10 % quantile of the match end positions. We excluded signatures that produced too many false-alarms, as too few bytes were specified in the signature. As can be seen, the majority of the signatures match below a dialog position of 2048 bytes. Of the Sourcefire ruleset, events of several rules matched after the limit of 2048 bytes. These rules specifically matched shellcode within data streams of various application protocols. Shellcode is usually embedded within binary data of exploits. These are often not directly embedded in the application layer protocol, but in bulk data that is transferred by the application protocol (e.g., within the mail body in SMTP).

The quality of DPA can be estimated by looking at the data reduction rate and corresponding analysis quality depicted in Figure 3: As the detection capabilities and amount of data reduction highly depends on the application-layer protocol, we plot the results for multiple applications separately. The x-axis defines the number of bytes captured after each dialog change. The upper diagram shows the ratio of aggregated DPA data to the amount of original data. The lower diagram displays the ratio of events that would have been detected by an IDS using the given DPA dialog length. Our graph clearly shows that a dialog length of 2000 bytes easily ensures that more than 98 % of events can still be detected, while reducing the amount of data to be analyzed to about 1 %.

The number of dialog segments in a connection is also highly dependent on the application protocol. Figure 4 shows an ECDF of the number of dialog segments in one connection. In HTTP, the most used protocol in our network, more than 70% of all connections contained exactly two dialog segments – one for the HTTP request, one for the HTTP response from the server. All HTTP connections with more than two dialog segments use pipelining, so multiple requests and responses are contained within a single TCP connection. This feature is included in most browsers nowadays and provides a significant speedup. Protocols, where the server sends a greeting message before the client issues a command like SMTP or the FTP control channel, show a strong preference to an uneven number of dialog segments. These two protocols also clearly show that multiple bidirectional data exchanges are needed in one connection for a successful communication, like login,

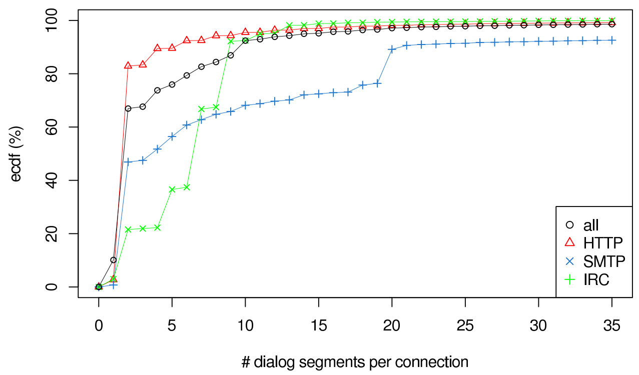

**Figure 4: ECDF of number of dialog segments per connection**

exchange of server features, and so on. In the example of SMTP, often up to 9 dialog segments are commonly used. Dialog segments are identified on the transport layer, not on the application layer. So there may be a discrepancy between dialogs on these different layers: multiple commands may be issued in one data chunk. Then this is visible as only one dialog segment in the transport layer and the number of total dialog segments is reduced. This can not be done in SSH, as almost every command issued for establishing the secure connection depends on the response of the previous command – the communication dialog between server and client can not be sped up like in other protocols. We were able to observe this in our experiments, as very few SSH connections contained between 2 to 9 dialog segments.

## 4. CONCLUSIONS

We introduced a new monitoring technique, which we call Dialog-based Payload Aggregation (DPA). It filters payload data for intrusion detection for individual connections. Almost all application protocols are based on bidirectional data exchange, which is usually visible in headers of the transport layer. So by observing the sequence counter of TCP streams, we identify changes of the data transfer direction and are able to extract security-relevant packet payload with a very lightweight algorithm.

According to our preliminary experimental results using data from a university network, this method allows to achieve detection of more than 98 % of security-relevant events with a data reduction of 99 % using the most popular IDS Snort with several common rulesets. These results show that our selection algorithm is very well suited for intrusion detection, as it mainly picks relevant data for the detection rules within the IDS. If we assume a linear relation between data rate and processing cost of an IDS, in theory, our method results in a speedup of around two magnitudes. Furthermore, DPA also introduces a new measure: the length of individual dialog segments. This measure may provide a base for application identification or anomaly detection algorithms. Current work is to implement the presented technique into our monitoring framework Vermont and perform performance tests in combination with the IDS Snort.

Future work includes the study of another phenomenon that cannot be detected by the current version of DPA: Malicious programs have the possibility to avoid detection by exploiting the semantic difference between application and transport layer: programs may send multiple control messages at once and exceed the filter limit of DPA. The server will then respond to all this messages in one go, as seen from the transport layer.

## 5. REFERENCES

[1] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer. Predicting the resource consumption of network intrusion detection systems. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):437–438, 2008.

[2] S. Kornexl, V. Paxson, H. Dreger, R. Sommer, and A. Feldmann. Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic. In *ACM IMC 2005*, pages 267–272, Berkeley, CA, October 2005. ACM.

[3] T. Limmer and F. Dressler. Flow-based Front Payload Aggregation. In *IEEE LCN 2009, WNM Workshop*, pages 1102–1109, Zurich, Switzerland, October 2009. IEEE.

[4] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis. Regular Expression Matching on Graphics Hardware for Intrusion Detection. In *RAID 2009*, pages 265–283, Saint-Malo, France, September 2009. Springer.