Log File Analysis with Context-Free Grammars

Gregory Bosman, Stefan Gruner

Chapter 10

LOG FILE ANALYSIS WITH CONTEXT-FREE GRAMMARS

Gregory Bosman and Stefan Gruner

Abstract Classical intrusion analysis of network log files uses statistical machine learning or regular expressions. Where statistically machine learning methods are not analytically exact, methods based on regular expressions do not reach up very far in Chomsky's hierarchy of languages. This paper focuses on parsing traces of network traffic using context-free grammars. "Green grammars" are used to describe acceptable log files while "red grammars" are used to represent known intrusion patterns. This technique can complement or augment existing approaches by providing additional precision. Analytically, the technique is also more powerful than existing techniques that use regular expressions.

Keywords: Intrusion detection, log file analysis, context-free grammars

1. Introduction

Most modern intrusion analysis systems rely on pattern recognition to differentiate malicious network traffic from benign traffic [1, 4, 6]. Anomaly-based detection systems are designed to recognize traffic generated during normal network operations. These systems use information such as source and destination addresses and ports, applications that generate traffic, and other header information to determine whether or not a packet should raise an alarm. Sometimes, the analysis proceeds one step further, scrutinizing the packets as a stream instead of just packet by packet. Signature-based analysis systems operate similarly and, in some cases, search for signatures of injected code in network packets. The search typically attempts to identify a series of bits that match certain regular expressions.

State-based analysis is a promising security approach [7, 10]. Gudes and Olivier [7] emphasize that the state of an application should also be

considered in assessing the security of an application. They reason that certain messages received by an application, while perfectly safe in one state, may compromise the security of the application should the messages be received in some other, more vulnerable state. Therefore, the state of an application should be taken into account when determining whether or not incoming data is harmful. Gudes and Olivier [7] proposed the use of context-free grammars for state-based analysis. Their approach entails the construction of a grammar to represent the transitions from state to state internally within an application; the grammar also describes the actions that are allowable in any given state. This paper presents an enhancement of the context-free grammar approach for the purpose of intrusion detection.

2. Related Work

Memon [5] has employed context-free grammars in a log file categorization system. Memon developed a grammar capable of representing a single packet in a data stream, including information such as packet protocol, source and destination. Also, a method of grammar inference was developed, which is capable of expanding an existing grammar modeled to identify benign packets at the discretion of an administrator. However, Memon's approach does not consider the sequencing of packets in a data stream, nor does it take into account the "statefulness" of an interaction between network applications. Also, a grammar that could capture the statefulness of applications communicating over a network would be considerably more complex than the grammars used by Memon.

The possibility of dealing with too many false alerts in automated intrusion detection systems was addressed by Harang and Guarino [3]. They used various heuristics to condense a massive amount of elementary alerts into a manageable number of meta-alerts. However the alert condensation technique has limited theoretical underpinnings. Similar work by Valdez and Skinner [9] resulted in a fusion system that groups alerts based on their similarity (near-matches) to existing meta alerts.

3. Method

The context-free grammar described in this paper handles a subset of malicious attack vectors. The grammar, which fits well into the digital forensics context, is related to formal methods in the field of model-based testing [2] and may be regarded as a limited form of automated verification on the basis of specific samples. A proof goal is a decision whether or not a network intrusion attempt has occurred in a specific situation.

```
TRACE      -> e | TRANSF | ADRESO | DNSQRY
TRACED1    -> TRANSFD1 | ADRESOD1 | DNSQRY1
TRACET1    -> TRANSF1  | ADRESOT1 | DNSQRYT1
TRACET2    -> TRANSF2  | ADRESOT2 | DNSQRYT2
TRACET3    -> PDU      | ADRESOT3 | DNSQRYT3
TRACET1D1  -> TRANSF1D1 | ADRESOT1D1 | DNSQRY1T1
TRACET2D1  -> TRANSF2D1 | ADRESOT2D1 | DNSQRY1T2
TRACET3D1  -> PDUD1  | ADRESOT3D1 | DNSQRY1T3
TRACEP1    -> PDU1   | ADRESOP1   | DNSQRYP1
TRACEP2    -> PDU2   | ADRESOP2   | DNSQRYP2
TRACEP3    -> PDU3   | ADRESOP3   | DNSQRYP3
TRACED1P1  -> PDU1D1 | ADRESOD1P1 | DNSQRY1P1
TRACED1P2  -> PDU2D1 | ADRESOD1P2 | DNSQRY1P2
TRACED1P3  -> PDU3D1 | ADRESOD1P3 | DNSQRY1P3
TRANSF     -> syn TRACET1
TRANSF1    -> synack acko get TRACET2
TRANSF2    -> acki TRACET3
TRANSFD1   -> syn TRACET1D1
TRANSF1D1  -> synack acko get TRACET2D1
TRANSF2D1  -> acki TRACET3D1
ADRESO     -> arpq arpr TRACE
ADRESOT1   -> arpq arpr TRACET1
ADRESOT2   -> arpq arpr TRACET2
ADRESOT3   -> arpq arpr TRACET3
```

Figure 1. Green grammar.

Two types of grammars are used to tackle the problem effectively from both sides:

- **Green Grammar:** One green grammar is used to describe log files corresponding to harmless communications. A log file that is successfully parsed with a green grammar is regarded as "hypothetically harmless."
- **Red Grammars:** Several red grammars are used to describe log files that contain traces corresponding to known malicious communications patterns. A log file that is successfully parsed with a red grammar is regarded as "harmful" and raises an alert. A log file that cannot be parsed with a red grammar is regarded as "hypothetically harmless."

Figures 1 and 2 present the green grammar that represents acceptable traffic. The grammar was generated from experimental traffic and log files created during the use of a web browser. Each terminal symbol in the grammar represents a single packet that was recorded. The grammar as a whole models a string of packets of a "conversation" that occurs

```
ADRESOT1D1 -> arpq arpr TRACET1D1
ADRESOT2D1 -> arpq arpr TRACET2D1
ADRESOT3D1 -> arpq arpr TRACET3D1
ADRESOP1   -> arpq arpr TRACEP1
ADRESOP2   -> arpq arpr TRACEP2
ADRESOP3   -> arpq arpr TRACEP3
ADRESOD1P1 -> arpq arpr TRACED1P1
ADRESOD1P2 -> arpq arpr TRACED1P2
ADRESOD1P3 -> arpq arpr TRACED1P3
DNSQRY     -> dnsq TRACED1
DNSQRY1    -> dnsr TRACE
DNSQRYT1   -> dnsq TRACET1D1
DNSQRYP1   -> dnsq TRACED1P1
DNSQRY1T1  -> dnsr TRACET1
DNSQRY1P1  -> dnsr TRACEP1
DNSQRYT2   -> dnsq TRACET2D1
DNSQRYP2   -> dnsq TRACED1P2
DNSQRY1T2  -> dnsr TRACET2
DNSQRY1P2  -> dnsr TRACEP2
DNSQRYT3   -> dnsq TRACET3D1
DNSQRYP3   -> dnsq TRACED1P3
DNSQRY1T3  -> dnsr TRACET3
DNSQRY1P3  -> dnsr TRACEP3
PDU        -> data TRACEP1 | http TRACEP3
PDU1       -> data TRACEP2 | http TRACEP3
PDU2       -> acko TRACET3
PDU3       -> acko TRACE
PDUD1      -> data TRACED1P1 | http TRACED1P3
PDU1D1     -> data TRACED1P2 | http TRACED1P3
PDU2D1     -> acko TRACET3D1
PDU3D1     -> acko TRACED1
```

Figure 2. Green grammar (continued).

when a web page is transferred. Ideally, the conversation follows the simplified pattern:

```
TRACE -> syn synack acko get acki PDU
PDU   -> data data acko PDU
       | data http acko
       | http acko
```

A communication involving a browser begins with a handshake, a sequence of packets with synchronization flags or acknowledgement flags or both flags set. Next, a request is made for data transfer, a GET followed by a packet in which the acknowledgement flag is set (again). Data transfer then begins. This is represented by the protocol data unit (PDU) section. The rest is the data transfer of two packets at a time until

Table 1. Green grammar results.

Input	Total	Accepted	Rejected	Lex Err	Syn Err
Benign	100	94	6	6	0
Malicious	100	0	100	28	72

the transfer is complete. After every two packets, an acknowledgement is sent to indicate there are no errors and that it is possible to continue with the transfer. When there is no more data, a final HTTP packet is received to indicate the end of the stream, followed by one last outgoing acknowledgement to the web server.

The example pattern is an ideal model of a conversation between a client and server. In practice, however, streams tend to be interspersed with additional mini-conversations that carry meta information. For example, an initial Address Resolution Protocol (ARP) message may be sent in order to find the location of a web page. Also, in nearly every trace, there is at least one request for a Domain Name Server (DNS), which tends to jump straight into the middle of the main conversation with the web server. Therefore, an appropriate green grammar must be constructed such that it retains the internal state of a conversation when it is interrupted and returns to the original state after the interrupt.

This implies that an entire conversation must complete in order to be recognized as a benign trace. If the conversation starts, it must terminate successfully. If it does not, then the browser may have been compromised (unless the communication was technically interrupted, for example by a browser crash on the client side). In such a case, the parsing of the corresponding trace leads to a syntax error and the trace is classified as "suspicious."

It is worth noting that the mini-ARP-conversation is always completed without an interruption. However, as in the case of a web page request, a DNS request is susceptible to interruption. Unfortunately, regular expressions (unlike the more expressive context-free grammars) are inadequate to remember the status of a conversation before the occurrence of an interrupt.

4. Experimental Results

We conducted an experiment that used the green grammar in a controlled laboratory network setting with various permissible and non-permissible input strings. ANTLR [8] was used to generate the parser for the green grammar. Table 1 summarizes the experimental results.

Two types of errors are of interest in the analysis:

- **Alpha Error (False Positive):** This error is the incorrect rejection of benign input (6/100 = 6% from Table 1). An alpha error is annoying or inconvenient, but it is not harmful.
- **Beta Error (False Negative):** This error is the incorrect acceptance of malicious input (0/100 = 0% from Table 1). A beta error is interpreted as being harmful.

Despite the zero beta error, a problem with the grammar is that not all parsing failures resulted from syntax errors, which would be the ideal case if the grammar was *a priori* aware of all possible input tokens. The non-zero lexical errors (errors prior to the parsing phase) show that an input stream can contain unknown symbols that the grammar does not recognize. Thus, the suitability (fit-for-purpose quality) of the intrusion detection grammar can be estimated as follows:

- In the ideal alpha case, no parsing errors should occur. Since all the errors occurred during the lexical analysis phase, 100% percent of the alpha errors are from not recognizing all the symbols in the input strings.
- In the ideal beta case, only parsing errors should occur. Only 72% percent of the (correct) rejection decisions were based on parsing whereas 28% percent of all rejections were based on "decisionless" lexical errors before the decision-making parsing phase. Therefore, the beta errors are estimated to be as high as 28%.

We conjecture that the errors appear in the context of the occasional mini-communications, which can be corrected for by modifying the manner in which the grammar is processed.

The experiments with malicious input streams were restricted to scenarios in which communications terminate mid-stream, as in the case of buffer overflow attacks. The parser should correctly recognize the absence of stream-closing data packets and label the stream as suspicious. However, in one case, the lexical analyzer could not recognize a termination symbol that was sent properly to terminate a conversation. This problem might be linked to speed discrepancies between data download and analysis. Specifically, the parsing unit may prematurely believe that an input stream has been "hacked" while in fact a harmless communication is in progress.

5. Discussion

Because context-free languages are a superclass of regular expressions and are, therefore, more complex, they have greater computational re-

quirements. A critical question is whether the additional complexity provided by context-free grammars renders the approach impractical. To answer this question, more examples that cannot be treated with regular expressions must be identified. The speed of analysis is the main reason for using regular expressions with Cisco networking devices because the analysis of data streams proceeds in an online and on-the-fly manner. Regular expression analysis can be implemented in a runtime-efficient manner. However, *ex post facto* log file analysis does not have strict real-time processing requirements.

A practical problem is posed by interrupted communication attempts, for example, due to a crashed or frozen browser on the client side of an Internet connection. On the server side, this results in a broken trace in the log file that cannot be parsed, regardless of whether the interrupted communication is malicious or harmless. Traces of interrupted communication attempts could increase the rate of alpha errors. To tackle this problem, a method is needed to divide a large trace into shorter sub-traces that could be parsed individually. Alternatively, the grammar could be modified so that interrupted communications are explicitly identified as such.

6. Conclusions

The application of context-free grammars to describe acceptable log files and intrusion patterns can significantly enhance log file analysis. The resulting approach is more powerful and provides better precision than existing techniques based on regular expressions.

Our future research will focus on developing a software framework that enables users to generate case-specific grammars and conduct detailed analyses. We will also attempt to cast a large number of well-known intrusion patterns as case-specific red grammars. A suitable combination of green and red grammars would reduce the alpha and beta error rates; however, this would require an additional meta decision policy for cases where the decisions made by the green and red parsers are inconsistent.

Acknowledgement

We thank Bruce Watson for his assistance with the regular expression technique in the context of Cisco networking equipment. We also thank the anonymous reviewers and the conference attendees for their insightful critiques and comments.

References

[1] S. Axelsson, Intrusion Detection Systems: A Survey and Taxonomy, Technical Report, Department of Computer Science, Chalmers University, Goteborg, Sweden, 2000.

[2] S. Gruner and B. Watson, Model-based passive testing of safety-critical components, in *Model-Based Testing for Embedded Systems*, J. Zander, I. Schieferdecker and P. Mosterman (Eds.), CRC Press, Boca Raton, Florida, pp. 453–483, 2011.

[3] R. Harang and P. Guarino, Clustering of Snort alerts to identify patterns and reduce analyst workload, *Proceedings of the 2012 Military Communications Conference*, 2012.

[4] T. Lunt, A survey of intrusion detection techniques, *Computers and Security*, vol. 12(4), pp. 405–418, 1993.

[5] A. Memon, Log File Categorization and Anomaly Analysis Using Grammar Inference, M.S. Thesis, School of Computing, Queen's University, Kingston, Canada, 2008.

[6] P. Ning and S. Jajodia, Intrusion detection techniques, in *The Internet Encyclopedia, Volume 2*, H. Bidogli (Ed.), Wiley, Hoboken, New Jersey, pp. 355–367, 2004.

[7] M. Olivier and E. Gudes, Wrappers: A mechanism to support state-based authorization in web applications, *Data and Knowledge Engineering*, vol. 43(3), pp. 281–292, 2002.

[8] T. Parr, *The Definitive ANTLR Reference: Building Domain-Specific Languages*, Pragmatic Bookshelf, Raleigh, North Carolina, 2007.

[9] A. Valdez and K. Skinner, Probabilistic alert correlation, *Proceedings of the Fourth International Symposium on Recent Advances in Intrusion Detection*, pp. 54–68, 2001.

[10] S. Zhang, T. Dean and S. Knight, A lightweight approach to state-based security testing, *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research*, article no. 28, 2006.