

# ModSecurity Audit Log Analysis

- There are 2 types of audit log formats
  - The Serial audit log type logs all data to one file (is the deprecated format)
  - The new Concurrent log type creates individual files for each request data
- It is highly recommended that the Concurrent log type be used for the following reasons
  - Increased performance of logging
  - Data be sent to the ModSecurity Console for analysis
- If you must use the Serial format, you will undoubtedly find that conducting audit log analysis and searches are cumbersome due to the fact that the records are multiline
- This makes it difficult to extract out the entire record of interest with standard Unix tools such as grep and egrep (*egrep -C 5 string*)
- There had to be a better way to search and extract out records...



ModSecurity currently supports 2 logging formats – Serial and Concurrent with the later being the most currently and preferred method. The data provided by the ModSecurity audit\_log file is invaluable when conducting incident response for web attacks. The data represents the entire client request including the client headers and data payloads. While the value of this data is undeniable, there are some practical issues to deal with if this data is to be used in an efficient manner. For those of you who have been using ModSecurity for some time, you have almost certainly run into this issue that I am about to discuss.

While using grep/egrep searches does show more data from the log entry, it does not perform well in bulk searches as there are never a consistent number of client headers being sent, so the accuracy of the returned data is a bit off. What I needed was a script that functioned similarly to egrep; however, it would be able to understand the ModSecurity audit\_log format and be able to extract out the multi-line record.

## Sgrep.pl Script for Log Analysis

- Perl to the rescue!
- A script was created called “sgrep.pl”
  - It is able to search the audit\_log for the specified search string
  - It uses the Perl line separator function to identify the “=====” record separator that ModSecurity uses for the Serial format
- This script is tremendously useful for parsing and extracting data from the standard ModSecurity audit\_log file
- Execute sgrep.pl with the “-h” flag for help syntax



This led to the creation of a script called sgrep.pl. Sgrep is a PERL script that accomplishes the tasks that we need to successfully parse and extract entire audit\_log records that contain the desired search text string.

# Sgrep.pl Help Menu

```
# ./sgrep.pl -h
This program does...
usage: $0 [-hf:s:v:]
    if a file is compressed then it will be uncompressed on the
    fly

    default    : display usage
    -f file    : file to search through
    -s string  : string to match on - enclosed in quotes if it
                  contains spaces
    -v level   : verbose output
    -h        : this (help) message

example:
    $0
    $0 -f
    $0 -s
    $0 -v level Verbose/Debug messages, where level = 0..9
```



Here is the help menu for the sgrep.pl script.

## Sgrep.pl Usage Example

```
# /tools/sgrep.pl -f audit_log -s "200.189.60.251"
=====
Request: 200.189.60.251 - - [18/May/2005:23:58:17 --0400] "GET /awstats/awstats
.pl?configdir=|echo%20;echo%20;id;echo%20;echo| HTTP/1.0" 403 743
Handler: cgi-script
-----
GET /awstats/awstats.pl?configdir=|echo%20;echo%20;id;echo%20;echo| HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0b; Windows NT 5.0)
Host: www.companyx.com
Via: 1.0 lilith.persistelecom.com.br:3128 (squid/2.5.STABLE8)
X-Forwarded-For: 10.1.1.9
Cache-Control: max-age=259200
Connection: keep-alive
ModSecurity-message: Access denied with code 403. Pattern match "\;id" at THE_REQUEST
ModSecurity-action: 403

HTTP/1.0 403 Forbidden
Connection: close
```



As you can see, `sgrep.pl` was able to extract out the entire record for this `awstats` command injection attempt from the specified IP address `200.189.60.251`.

# Pipelining Sgrep Commands

```
# ./sgrep.pl -f audit_log -s "ccbill" | ./sgrep.pl -f - -s 68.16.164.147
=====
Request: 68.16.164.147 - - [Thu Mar 11 15:29:49 2004] "GET
  http://www.access.ccbill.com/jettis/add-passwd-old.cgi HTTP/1.0" 500 454
Handler: proxy-server
Error: The proxy server could not handle the request <EM><A
  HREF="http://www.access.ccbill.com/jettis/add-passwd-
  old.cgi">GET&nbsp;http://www.access.ccbill.com/jettis/add-passwd-
  old.cgi</A></EM>.<P>
Reason: <STRONG>Host not found</STRONG>
-----
GET http://www.access.ccbill.com/jettis/add-passwd-old.cgi HTTP/1.0
Cache-Control: no-cache
Connection: close
Host: www.access.ccbill.com
Pragma: no-cache
Proxy-Connection: keep-alive
Referer: http://www.access.ccbill.com/jettis/add-passwd-old.cg
```



Sometimes, the search string used with Sgrep produces too many results. In order to narrow down the results and/or make a more complex search, you can use multiple sgrep.pl searches together. It is possible to take the output of one sgrep.pl search and use it as the input to another search. This is accomplished by using the Unix pipe character “|” to join 2 separate sgrep.pl commands together. On the second sgrep.pl command, you must specify a dash “-” character to the “-f” file flag. This tells Perl to use standard input instead of reading an actual file.