ALGORITHME SOLUTIONS INC.

Security analysis: logging

by :
Donavan Martin
Candidate to the engineering profession (CEP at oiq.qc.ca)

Public version
Translated with DeepL.com
Saint-Denis-de-Brompton – 14 juin 2021

Table des matières

1 Introduction

Knowing the offensive techniques is the best way to protect yourself from computer attacks. In the field of cybersecurity, an excellent logging allows to collect a lot of information on the techniques used during the attempts. The analysis of intrusion attempts, injection and information gathering is essential to defend a computer system. In this publication, we discuss log analysis concepts with concrete examples as a way to raise awareness of cybersecurity.

2 Logging

Depending on the server configuration, the logs can be located in several places. Indeed, the process configuration usually allows to move the log file to a different directory than the default one. In the following examples, the default configuration of an Ubuntu server is used.

2.1 Composition of a log entry

Conventionally, the entry in a log includes a date, time, user, process, status and description about the event.

```
<Date><Time><User><Process><Etat><Event>
```

2.2 Example of process log analysis

The list of default process logs and log directories can be displayed using the following command :

```
ls -lt /var/log/
```

Let's take an example to analyze the last 10 syslog entries.

```
tail -10 /var/log/syslog
Jun  9 00:23:03  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=92.118.161.1 DST=143.110.xxx.xxx <...>
Jun  9 00:23:43  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=45.134.26.50 DST=143.110.xxx.xxx <...>
Jun  9 00:24:14  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=159.89.150.163 DST=143.110.xxx.xxx <...>
Jun  9 00:24:37  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=45.134.26.56 DST=143.110.xxx.xxx <...>
Jun  9 00:24:38  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=103.145.13.26 DST=143.110.xxx.xxx <...>
Jun  9 00:24:38  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=89.248.165.104 DST=143.110.xxx.xxx <...>
Jun  9 00:24:56  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=181.214.206.101 DST=143.110.xxx.xxx <...>
Jun  9 00:25:16  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=45.143.203.16 DST=143.110.xxx.xxx <...>
Jun  9 00:25:37  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=45.134.26.55 DST=143.110.xxx.xxx <...>
Jun  9 00:26:03  <...> [UFW BLOCK] IN=eth0 OUT= MAC=b2:a5:8a:27:65:6e:fe:<...> SRC=162.218.65.219 DST=143.110.xxx.xxx <...>
```

We can quickly observe that there is a problem with the banning. In fact, the firewall does not ban the MAC address and no other process bans these intrusion attempts. Note that Fail2Ban is used, but banning by MAC address is not a Fail2Ban feature. We must therefore find the source of the problem. Since these logs look like a brute force attempt, we can analyze the authentication file :

```
tail -10 /var/log/auth.log
Jun  9 10:53:51 <...> sshd[97108]: Failed password for invalid user vagrant from 209.141.37.3 port 35568 ssh2
Jun  9 10:53:52 <...> sshd[97107]: Failed password for invalid user ubuntu from 209.141.37.3 port 35562 ssh2
Jun  9 10:53:52 <...> sshd[97104]: Failed password for root from 209.141.37.3 port 35560 ssh2
Jun  9 10:53:52 <...> sshd[97106]: Failed password for invalid user test from 209.141.37.3 port 35564 ssh2
Jun  9 10:53:52 <...> sshd[97105]: Failed password for invalid user user from 209.141.37.3 port 35566 ssh2
Jun  9 10:54:46 <...> wordpress(<domain>)[96225]: Authentication failure for user1 from 172.70.98.50
Jun  9 10:55:00 <...> wordpress(<domain>)[96226]: Authentication failure for user2 from 172.69.34.238
```

The authentication log shows two processes that generate log entries with intrusion attempts, sshd and wordpress. Finally, from this information we can increase the security by configuring the two processes.

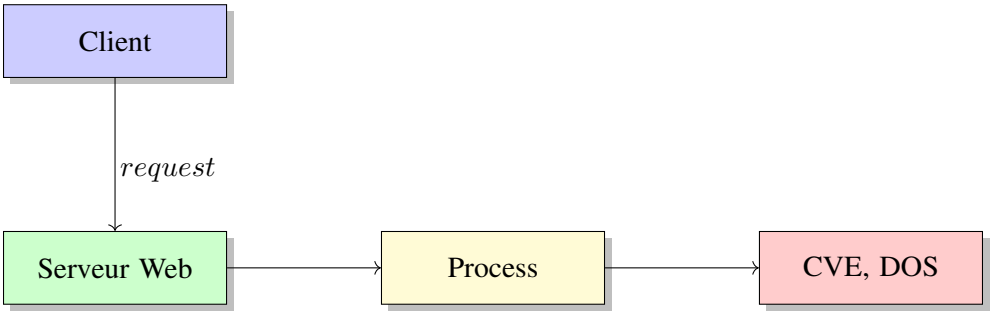

FIGURE 2-1 – Example of a process log analysis diagram

2.3 Example of query analysis logs

The two most used types of requests are GET and POST. The GET type takes parameters (called params) in the url. For example, the request GET *www.google.ca ?cookie=123* includes the parameter cookie whose value is 123. Without limitation, the response of a GET request can be an html page, a string or a JSON object. In general, the GET request is used to retrieve information from the state of the *params*.

By convention, the POST request is generally used to add, modify and delete information on the server. To do this, the POST request generally uses the *body* of the request to send information to the server.

One of the problems that can occur with these two types of requests is injection. If the parameters or the body are not parsed correctly, then injection is likely. A simple solution to validate the params or the body is to add a data schema validator. Let's take an example with an AJAX request code on a Wordpress server :

```
add_action( 'wp_ajax_endpoint', 'save_secret_info');
function save_secret_info() {
  if ( !is_admin()
     || !isset($_POST['secret'])
     || !wp_verify_nonce($_POST['secret'], 'algosol-nonce')) {
    $output = "Permission denied: nonce invalid";
     log_ajax($output );
    wp_die("permission denied");
  }else {
```

```
    if(is_string($_POST['secret'])){
      // A more secure way to save secret
      log_ajax("saved" );
      wp_die("saved");
    }else{
      log_ajax("secret n'est pas une chaine de caracteres");
      wp_die("invalid format");
    }
  }
}

function log_ajax($message){
 $date = date('Y-m-d H:i:s');
 $user = get_current_user ( );
 $process = "AJAX";
 error_log( $date. " ". $user. " ". $process." ". $message);
}
```

According to the previous code, each validation condition of the query is recorded in the logs. This example demonstrates the basis of a schema-based validation, but this example could be much more precise.

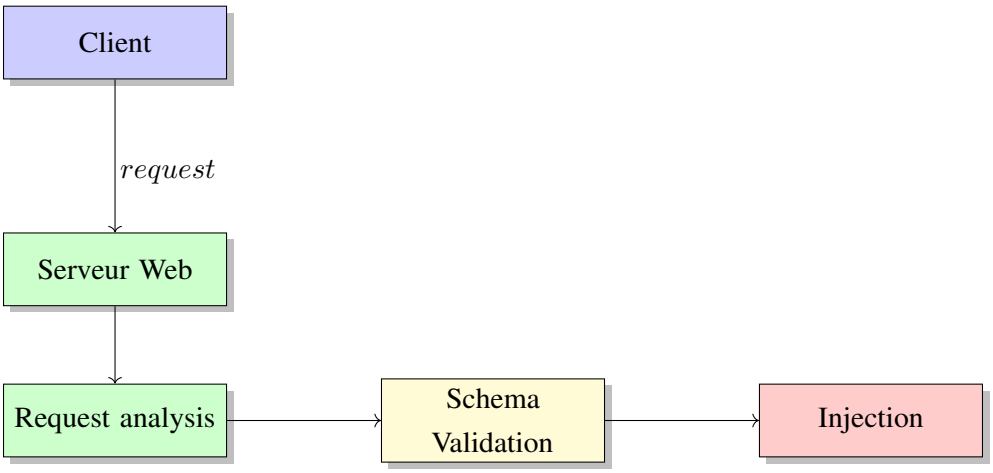

FIGURE 2-2 – Example of a query log analysis scheme

2.4 Example of stack trace logs

When a malicious request is sent to the server, many conditions are to be expected. It may happen that some conditions are not validated by the processes and the request analysis stage. In this case, the execution of the request may lead to undesirable program behavior. If the source code of the program is not known to the attacker, then the attempts will have the form of probability. For example, the success of a Shellcode sent in a request is more difficult due to execution constraints and therefore it is more likely that several attempts will be necessary.

When a badly parsed request is executed by a computer system, the execution of the request frequently generates execution errors. The execution errors are usually displayed in the stack trace and the programming languages also allow to generate the stack trace or to generate an exception.

Once again, logs can help to detect this type of event. Besides, the stack trace often includes enough information to correct the problem by a programmer.

Here is an example of stack trace display command :

```
var_dump(debug_backtrace());
```

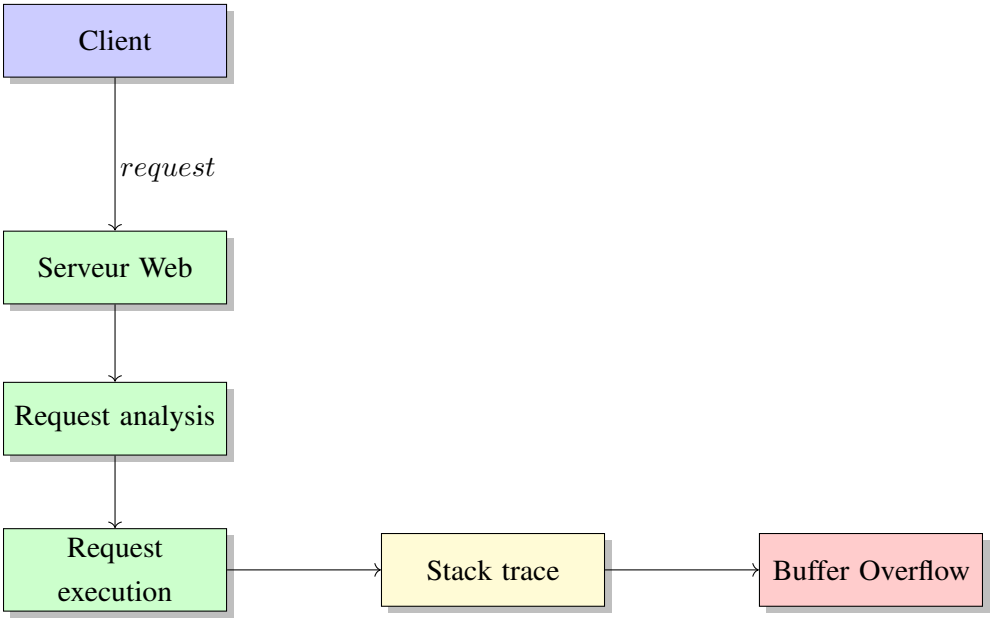

FIGURE 2-3 – Example of a query execution analysis scheme

2.5 Example of response logs

As mentioned earlier, the two most commonly used types of requests are GET requests and POST requests. When accessing a web page we use the GET request which includes state parameters. In addition, it is possible to determine the programs used by the web service. Moreover, free online tools exist. More concretely, the response sent by the server includes a lot of information.

For GET requests, it is enough to inspect the page. For example, it is possible to know if a website is designed with Wordpress by searching for the prefix "wp-". The following figure shows 76 results for the prefix "wp-". So, the probability that this site is written with Wordpress is very high.

Note that it is possible to analyze other types of requests. For example, we can look at the response headers to see if Cloudflare is being used as a proxy. The figure refcloudflare shows an example of a response for a POST request type.

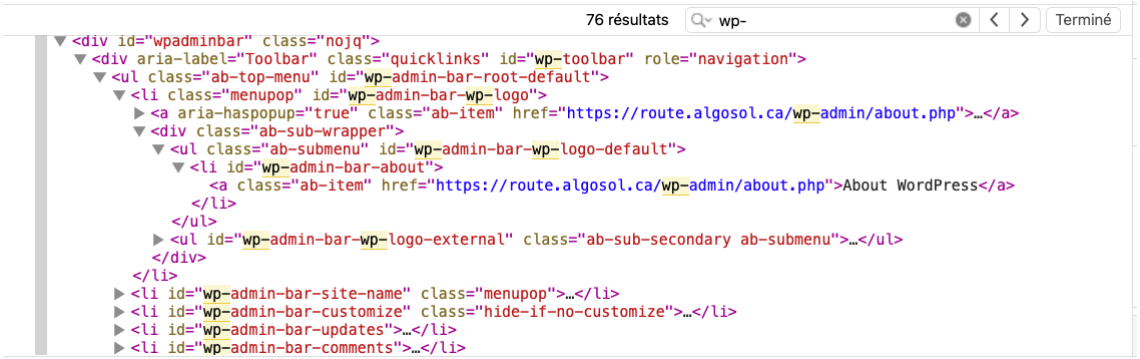

FIGURE 2-4 – Example of GET response analysis

En-têtes de réponse

Nom	Valeur
Content-Encoding	br
Cache-Control	no-cache, must-revalidate, max-age=0
Access-Control-Allow-Origin	https://route.algosol.ca
Referrer-Policy	strict-origin-when-cross-origin
Vary	Accept-Encoding
Date	Wed, 09 Jun 2021 13:15:20 GMT
Expires	Wed, 11 Jan 1984 05:00:00 GMT
Access-Control-Allow-Credentials	true
Content-Type	text/html; charset=UTF-8
X-Frame-Options	SAMEORIGIN
X-Content-Type-Options	nosniff
Server	cloudflare
x-robots-tag	noindex

FIGURE 2-5 – Example of POST response analysis

3 Relationship between programming, journals and identification

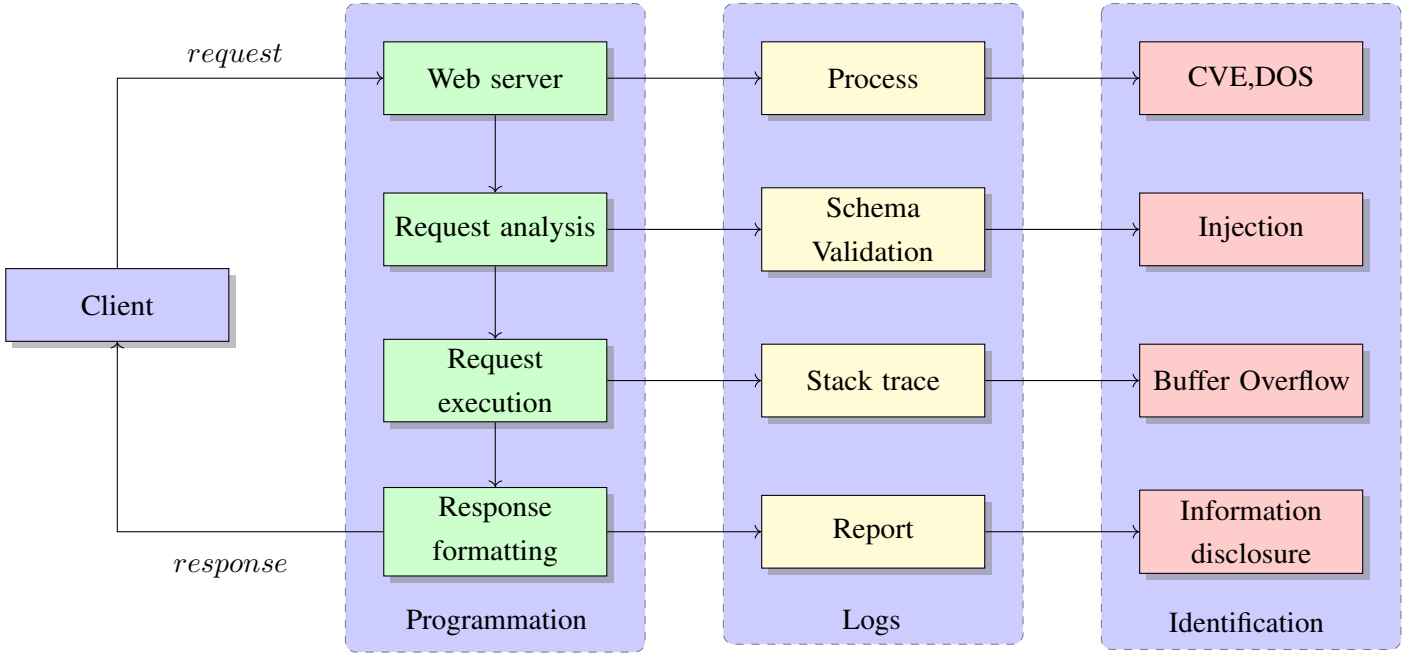

FIGURE 3-1 – Example of a secure log analysis scheme

4 Conclusion

This publication brings together fundamental analysis concepts for the protection of computer systems. Indeed, event logging is a goldmine of information about intrusion attempts and misuse of computer systems. This publication is only an introduction to log event analysis, as there are more effective ways to automatically analyze log events. Knowing the offensive techniques is the best way to protect yourself from computer attacks.