wordpress wp visitor statics Foreground secondary encoding injection vulnerability

Vulnerability Overview

In WP Visitor Statistics 6.9.3, due to simple filtering of user data, secondary encoding injection can be performed and directly spliced into sql statements, resulting in sql injection vulnerability.

Impact Version

Osamaesh WP Visitor Statistics<=6.9.3

Project Address: https://wordpress.org/plugins/wp-stats-manager/

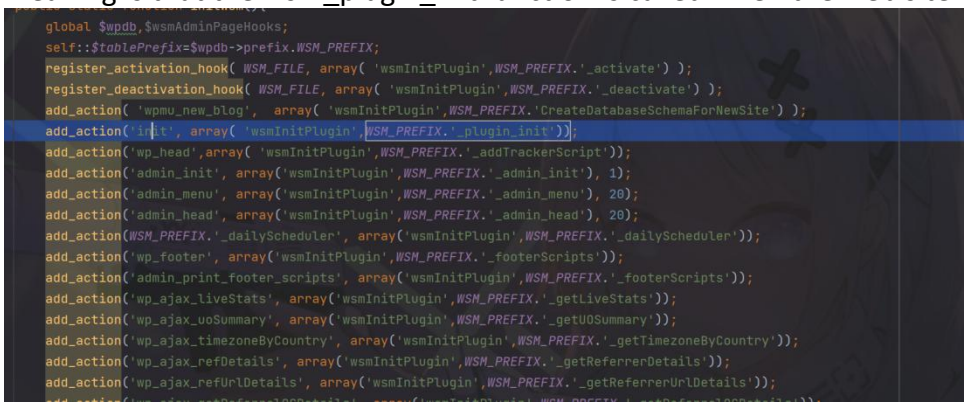
vulnerability recurrence

http://localhost/?wmcAction=wmcTrack&visitorId=121212%27+and+sleep(10)+or+%27

vulnerability analysis

The entry function of the plug-in is as follows

In the initWsm initialization function, the following actions are registered. The meaning is that the wsm_plugin_init function is called when the web site is visited



The wsm_plugin_init is as follows. When visiting a URL like http://xxxx/? Enter wsmRequests if wmcAction=wmcTrack

```
no usages
static function wsm_plugin_init() {
    global $wsmRequestArray;

//exit();
    if(isset($wsmRequestArray['wmcAction']) && ($wsmRequestArray['wmcAction']=='wmcTrack' || $wsmRequestArray['wmcAction']=='wm
        self::$objWsmRequest= new wsmRequests($wsmRequestArray);
    }

    load_plugin_textdomain( WSM_PREFIX, false, dirname(plugin_basename(WSM_FILE)).'/languages/' );
    self::$objStats=new wsmStatistics();
    self::$objDatabase=self::$objStats->wsm_getDatabaseObject();
```

In the constructor of wsmRequests, due to the simple filtering of user data, you can perform secondary encoding injection to adjust fnHandleVisit

```
}
$this->arrOriginal=$requests;
$this->objDatabase= new wsmDatabase();
$action=$this->fnGetParam( name: 'wmcAction');
if($action==='wmcTrack'){
    $this->fnSetLastHitTime();

    if(isset($requests['visitorId']) && $requests['visitorId'] !=''){
        $this->fnHandleVisit();
    }
    // header('Content-type: ' . "image/jpeg");
    die;
}else if($action==='wmcAutoCron'){
    wsmInitPlugin::wsm_fnCreateImportantViews();
    //$this->fnGenerateDailyReports();
    die;
}
```

The parameters accessed for the first time are processed in fnHandleNewVisit

```
function fnHandleVisit(){
    $this->fnSetTheNewProperties();
    $isNewVisit=$this->fnIsVisitNew();
    if($isNewVisit){
        $this->fnHandleNewVisit();
    }else{
        $this->fnHandleExistingVisit();
    }
}
2 usages
```

In fnInsertNewVisit, the passed-in visitorId is spliced and the sql statement is executed

```
function fnInsertNewVisit($properties){

    $fields=implode( separator: ',',$this->arrInsertLogVisit);
    $sql  = "INSERT INTO {$this->tablePrefix}{$this->arrTables['LOG_VISIT']} ($fields) VALUES (";
    if(isset($properties['visitId']) && $properties['visitId']!=0){
        foreach($this->arrInsertLogVisit as $key){
            if($key=='visitorId'){
                $sql.="'".$properties[$key]."',";
            }else if(is_numeric($properties[$key]) || $properties[$key]=='0'){
                $sql.=$properties[$key].',';
            }else{
                $sql.="'".addslashes($properties[$key])."',";
            }
        }

        $sql=rtrim($sql, characters: ',').')';
        //echo '<br>'.$sql;
        $this->wsmDB->query($sql);
        $this->fnLogError($properties);
        return intval($this->wsmDB->insert_id);
```

repair mode

The vendor has not released a solution yet

Safety researcher Mr. Li personally suggested:

Use parameterized queries or prepared statements: Rather than just using escape processing, a better approach is to use parameterized queries or prepared statements to build SQL queries. Parameterized queries are executed by separating the query statement from the parameters, thus separating the input data from the query logic and effectively preventing SQL injection attacks.