BOLETIM DE SEGURANÇA

Falha de crítica encontrada em plug-in LayerSlider do WordPress

heimdall
security research
A DIVISION OF ISH

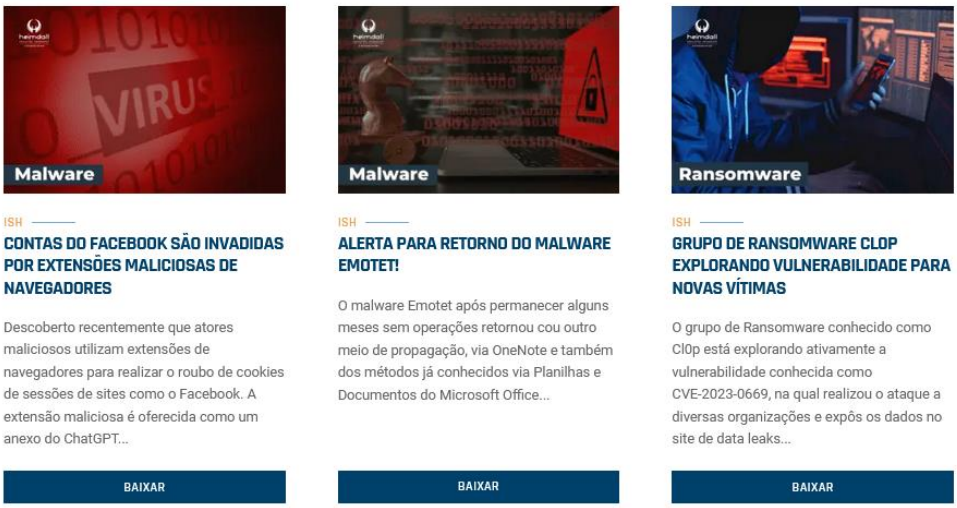
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

Heimdall Security Research

Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

Boletins de Segurança – Heimdall



SUMÁRIO

LISTA DE FIGURAS

1 SUMÁRIO EXECUTIVO

Em março de 2024, durante o segundo Bug Bounty Extravaganza, o pesquisador de segurança AmrAwad reportou uma vulnerabilidade de *SQL injection* CVE-2024-2879 categorizada como crítica. Essa falha está presente nas versões 7.9.11 e 7.10.0 e ocorre devido à falta de escape adequado no parâmetro fornecido pelo usuário e à preparação insuficiente da consulta SQL não autenticada no LayerSlider, podendo ser aproveitada para extrair dados confidenciais do banco de dados, como hashes de senha. Este plugin do WordPress tem mais de 1.000.000 de instalações ativas estimadas.

2 ANÁLISE SOBRE A VULNERABILIDADE

A funcionalidade de consulta de marcação pop-up do plug-in foi implementada de maneira insegura, possibilitando a injeção de SQL. Uma análise do código indica que o plugin emprega a função *ls_get_popup_markup()* para realizar a consulta da marcação dos controles deslizantes para pop-up. Nessa função, o 'id' pode ser definido por meio do parâmetro 'id'.

```
function ls_get_popup_markup() {

    $id     = is_numeric( $_GET['id'] ) ? (int) $_GET['id'] : $_GET['id'];
    $popup = LS_Sliders::find( $id );

    if( $popup ) {
        $GLOBALS['lsAjaxOverridePopupSettings'] = true;
        $parts  = LS_Shortcode::generateSliderMarkup( $popup );
        die( $parts['container'].$parts['markup'].'<script>'.$parts['init'].'</script>' );
    }

    die();
}
```

Figura 1 – Função ls_get_popup_markup().

Caso o parâmetro 'id' não seja numérico, ele é encaminhado sem qualquer sanitização para a função find() na classe LS_Sliders. Esta função realiza a consulta dos controles deslizantes de uma maneira específica se $args não for um número, uma string ou um array inteiro especial.

```
public static function find( $args = [] ) {

$userArgs = $args;

// Find by slider ID
if(is_numeric($args) && intval($args) == $args) {
    return self::_getById( (int) $args );

// Random slider
} elseif($args === 'random') {
    return self::_getRandom();

// Find by slider slug
} elseif(is_string($args)) {
    return self::_getBySlug($args);

// Find by list of slider IDs
} elseif(is_array($args) && isset($args[0]) && is_numeric($args[0])) {
    return self::_getByIds($args);

// Find by query
} else {


    // Defaults
    $defaults = [
        'columns' => '*',
        'where' => '',
        'exclude' => ['removed'],
        'orderby' => 'date_c',
        'order' => 'DESC',
        'limit' => 30,
        'page' => 1,
        'groups' => false,
        'data' => true,
        'drafts' => false
    ];

    // Merge user data with defaults
    foreach( $defaults as $key => $val ) {
        if( ! isset( $args[ $key ] ) ) {
            $args[ $key ] = $val;
        }
    }

    // Escape user data
    foreach( $args as $key => $val ) {
        if( $key !== 'where' ) {
            $args[ $key ] = esc_sql( $val );
        }
    }
```

Figura 2 – Encaminhamento do parâmetro para função find().

Todos os valores de $args são processados pela função esc_sql(), com a única exceção sendo o valor 'where'.

```
// Where
$where = '';
if(!empty($args['where']) && !empty($args['exclude'])) {
    $where = "WHERE ({$args['exclude']}) AND ({$args['where']}) ";

} elseif(!empty($args['where'])) {
    $where = "WHERE {$args['where']} ";

} elseif(!empty($args['exclude'])) {
    $where = "WHERE {$args['exclude']} ";
}


$sliders = $wpdb->get_results("
    SELECT SQL_CALC_FOUND_ROWS {$args['columns']}
    FROM $table
    $where
    ORDER BY `{$args['orderby']}` {$args['order']}, name ASC
    LIMIT {$args['limit']}

", ARRAY_A);
```

Figura 3 – Processamento dos valores $args.

A cláusula WHERE é incorporada à consulta sem a utilização da função prepare() do wpdb do WordPress. Essa função, se utilizada, parametrizaria e escaparia a consulta SQL para uma execução segura no WordPress, oferecendo proteção contra ataques de injeção de SQL.

Dada a estrutura da consulta, a injeção SQL baseada em união não é viável. Portanto, um atacante teria que adotar uma estratégia cega baseada no tempo para obter informações do banco de dados. Isso implicaria no uso de instruções *SQL CASE* em conjunto com o comando SLEEP(), monitorando o tempo de resposta de cada solicitação para extrair informações do banco de dados. Embora seja um método complexo, é frequentemente eficaz para explorar vulnerabilidades de injeção SQL e obter informações de um banco de dados.

A estrutura da consulta restringe a superfície de ataque a um método baseado no tempo, onde um invasor teria que monitorar o tempo de resposta de cada pedido para extrair informações do banco de dados.

3 RECOMENDAÇÕES

Abaixo são elencadas pela ISH, medidas que poderão ser adotadas visando a mitigação da referida ameaça, como por exemplo:

Parametrização das consultas

- Utilize consultas parametrizadas para garantir que os comandos SQL sejam interpretados corretamente pelo banco de dados, evitando a execução de comandos maliciosos.

Uso de stored procedures

- As stored procedures podem ajudar a evitar a injeção de SQL, pois permitem que o banco de dados trate as entradas como valores literais e não como código.

Higienização das entradas do usuário

- Certifique-se de que todas as entradas do usuário sejam verificadas e limpas antes de serem usadas em consultas SQL.

Restrição de procedimentos do banco de dados

- Limite o acesso aos procedimentos do banco de dados para minimizar o risco de exploração por um atacante.

Monitoramento de tráfego

- Utilize ferramentas de monitoramento de tráfego, como o WAF da Cloudflare, para detectar possíveis explorações de SQL.

Escapar entradas fornecidas pelo usuário

- Certifique-se de que todas as entradas fornecidas pelo usuário sejam escapadas corretamente para evitar a injeção de SQL.

Limitar privilégios de acesso

- Assegure-se de que as conexões ao banco de dados sejam feitas no contexto de um usuário com privilégios mínimos necessários.

Ocultar meta-caracteres injetados

- Use procedimentos armazenados para ocultar os meta-caracteres injetados.

4 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **Wordfance**
- **Bleepingcomputer**
- **NVD**

heimdall
security research
A DIVISION OF ISH