بسمه تعالی

آسیب پذیری بحرانی در سیستم مدیریت محتوای دروپال

۱ معرفی آسیب پذیری

به تازگی یک آسیب پذیری بحرانی در هسته‌ی سیستم مدیریت محتوای دروپال با درجه اهمیت بسیار حیاتی شناسایی و در تاریخ ۲۸ مارس اعلام شده است. این آسیب پذیری به نفوذگر امکان اجرای کدهای مخرب از راه دور را در نسخه های ۷ و۸ دروپال بدون نیاز به احراز هویت می‌دهد و نفوذ گران براحتی می‌توانند تمامی وب‌سایت شما را در اختیار بگیرند. این آسیب‌پذیری که با شناسه CVE-2018-7600 شناخته می‌شود بر روی هسته و نسخه‌های 7.x, 8.3.x, 8.4.x و 8.5.x دروپال تاثیر می‌گذارد.

۲ راهکار مقابله

- اگر از نسخه‌ی 7.x استفاده می کنید آن را به Drupal 7.58 ارتقا دهید. چنانچه در حال حاضر ارتقا به این نسخه امکان پذیر نمی‌باشد از وصله‌ی امنیتی استفاده نمایید. پیشنهاد می‌شود در اسرع وقت حتما نسخه‌ی دروپال خود را آپدیت نمایید.

https://www.drupal.org/project/drupal/releases/7.58

https://cgit.drupalcode.org/drupal/rawdiff/?h=7.x&id=2266d2a83db50e2f97682d9a0fb8a18e2722cba5

- چنانچه از نسخه‌ی 8.5.x استفاده می‌کنید آن را با نسخه‌ی ۸,۵,۱ جایگزین کنید. همچنین اگر در حال حاضر امکان تغییر نسخه‌ی دروپال فعلی موجود نمی‌باشد از وصله‌ی امنیتی استفاده کنید. پیشنهاد می‌شود در اولین فرصت از نسخه ی ۸,۵,۱ استفاده کنید.

https://www.drupal.org/project/drupal/releases/8.5.1

https://cgit.drupalcode.org/drupal/rawdiff/?h=8.5.x&id=5ac8738fa69df34a0635f0907d661b509ff9a28f

- نسخه‌های دروپال 8.3.x و 8.4.x دیگر پشتیبانی نمی‌شوند و وصله‌های امنیتی برای نسخه‌ی جزیی ارائه نمی‌شود اما با توجه به حیاتی بودن این باگ امنیتی پچ امنیتی برای این نسخه ها نیز ارایه شده است که حاوی اصلاحیه‌هایی برای سایت‌هایی است که تا به حال به نسخه‌ی ۸,۵,۰ به روز رسانی نشده اند.
- چنانچه از نسخه‌ی 8.3.x استفاده می کنید آن را به دروپال 8.3.9 ارتقا دهید و یا از این وصله‌ی امنیتی استفاده کنید.

https://www.drupal.org/project/drupal/releases/8.3.9

https://cgit.drupalcode.org/drupal/rawdiff/?h=8.5.x&id=5ac8738fa69df34a0635f0907d661b509ff9a28f

- چنانچه از نسخه‌ی امنیتی 8.4.x استفاده می‌کنید آن را به نسخه‌ی امنیتی 8.4.6 ارتقا دهید و یا از این وصله‌ی امنیتی استفاده کنید:

https://www.drupal.org/project/drupal/releases/8.4.6

https://cgit.drupalcode.org/drupal/rawdiff/?h=8.5.x&id=5ac8738fa69df34a0635f0907d661b509ff9a28f

اگر شما از هرکدام از این نسخه‌های دروپال دروپال ۸ قبل از 8.2.x استفاده می‌کنید ابتدا به نسخه جدیدتر ارتقا دهید و دستور العمل‌های بالا را دنبال کنید.

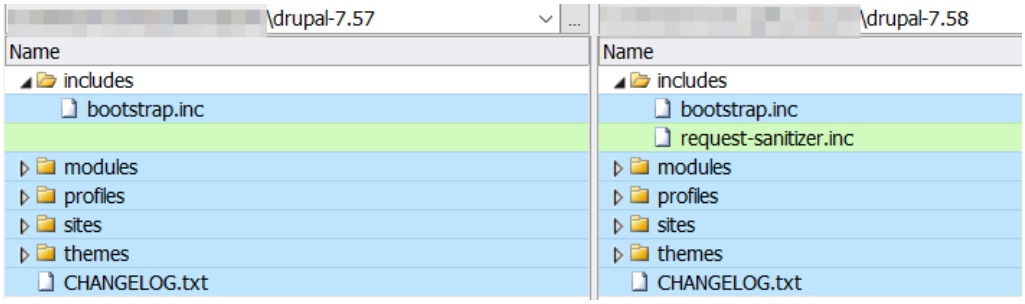
۳ بررسی فنی آسیب پذیری

آسیب پذیری بسیار بحرانی اجرای راه دور کد، مربوط به زیر سیستم‌های چندگانه دروپال است. این آسیب پذیری در هسته دروپال قرار دارد، و بدان معنی است که تمام محتوای دروپال، صرف نظر از هر پلاگین نصب شده، آسیب پذیر هستند.

محققان امنیتی جزئیات کامل مربوط به آسیب پذیری را ذکر نکرده و هیچ سوءاستفاده عمومی در دسترس هنوز مشخص نشده است. با این حال، با توجه به ماهیت متن باز دروپال، محققان امنیتی قادر به درک زمینه تغییر با استفاده از سوابق commit های Git هستند. تغییرات کد نشان می‌دهد که یک کتابخانه به نام request-sanitizer.inc به کد اضافه شده است. تابع اصلی در این کتابخانه "stripDangerousValues" نامیده می‌شود. بر این اساس دروپال دارای ضعف‌های زیادی در برابر ورودی‌های کاربر است.

با بررسی نسخه ۷٫۵۸ دروپال، مشخص می‌شود که یک فایل جدید اضافه شده و یک فایل به روز شده است.

دایرکتوری " /includes" حاوی چندین فایل inc. است که هنگام دسترسی به متغیرهای جانبی سرور و دستکاری داده های ارائه شده توسط کاربر در سرور، فراخوانی می‌شوند. در نسخه ۷٫۵۸ یک فایل جدید به نام request-sanitizer.inc اضافه شده است که حاوی توابع برای تمیز کردن ورودی کاربر از طریق GET، POST یا کوکی می باشد.



در فایل core/lib/Drupal/Core/DrupalKernel.php موارد مشخص شده در زیر، تغییر یافته‌اند:

```
@@ -20,6 +20,7 @@
 use Drupal\Core\Http\TrustedHostsRequestFactory;
 use Drupal\Core\Installer\InstallerRedirectTrait;
 use Drupal\Core\Language\Language;
+use Drupal\Core\Security\RequestSanitizer;
 use Drupal\Core\Site\Settings;
 use Drupal\Core\Test\TestDatabase;
 use Symfony\Cmf\Component\Routing\RouteObjectInterface;
@@ -542,6 +543,12 @@ public function loadLegacyIncludes() {
     * {@inheritdoc}
     */
   public function preHandle(Request $request) {
+    // Sanitize the request.
+    $request = RequestSanitizer::sanitize(
+      $request,
+      (array) Settings::get(RequestSanitizer::SANITIZE_WHITELIST, []),
+      (bool) Settings::get(RequestSanitizer::SANITIZE_LOG, FALSE)
+    );

     $this->loadLegacyIncludes();
```

و فایل core/lib/Drupal/Core/Security/RequestSanitizer.php با کدهای زیر اضافه شده است:

```
+<?php
+
+namespace Drupal\Core\Security;
+
+use Symfony\Component\HttpFoundation\Request;
+
+/**
+ * Sanitizes user input.
+ */
+class RequestSanitizer {
+
+  /**
+   * Request attribute to mark the request as sanitized.
+   */
+  const SANITIZED = '_drupal_request_sanitized';
+
+  /**
+   * The name of the setting that configures the whitelist.
+   */
+  const SANITIZE_WHITELIST = 'sanitize_input_whitelist';
+
+  /**
+   * The name of the setting that determines if sanitized keys are logged.
+   */
+  const SANITIZE_LOG = 'sanitize_input_logging';
+
+  /**
+   * Strips dangerous keys from user input.
+   *
+   * @param \Symfony\Component\HttpFoundation\Request $request
+   *   The incoming request to sanitize.
+   * @param string[] $whitelist
+   *   An array of keys to whitelist as safe. See default.settings.php.
+   * @param bool $log_sanitized_keys
+   *   (optional) Set to TRUE to log an keys that are sanitized.
+   *
+   * @return \Symfony\Component\HttpFoundation\Request
+   *   The sanitized request.
+   */
+  public static function sanitize(Request $request, $whitelist, $log_sanitized_keys = FALSE) {
```

```
+   if (!$request->attributes->get(self::SANITIZED, FALSE)) {
+     // Process query string parameters.
+     $get_sanitized_keys = [];
+     $request->query->replace(static::stripDangerousValues($request->query->all(), $whitelist, $get_sanitized_keys));
+     if ($log_sanitized_keys && !empty($get_sanitized_keys)) {
+       trigger_error(sprintf('Potentially unsafe keys removed from query string parameters (GET): %s', implode(', ', $get_
+     }
+
+     // Request body parameters.
+     $post_sanitized_keys = [];
+     $request->request->replace(static::stripDangerousValues($request->request->all(), $whitelist, $post_sanitized_keys));
+     if ($log_sanitized_keys && !empty($post_sanitized_keys)) {
+       trigger_error(sprintf('Potentially unsafe keys removed from request body parameters (POST): %s', implode(', ', $pos
+     }
+
+     // Cookie parameters.
+     $cookie_sanitized_keys = [];
+     $request->cookies->replace(static::stripDangerousValues($request->cookies->all(), $whitelist, $cookie_sanitized_keys)
+     if ($log_sanitized_keys && !empty($cookie_sanitized_keys)) {
+       trigger_error(sprintf('Potentially unsafe keys removed from cookie parameters: %s', implode(', ', $cookie_sanitized
+     }
+
+     if (!empty($get_sanitized_keys) || !empty($post_sanitized_keys) || !empty($cookie_sanitized_keys)) {
+       $request->overrideGlobals();
+     }
+     $request->attributes->set(self::SANITIZED, TRUE);
+   }
+   return $request;
+ }
+
+ /**
+  * Strips dangerous keys from $input.
+  *
+  * @param mixed $input
+  *   The input to sanitize.
+  * @param string[] $whitelist
+  *   An array of keys to whitelist as safe.
+  * @param string[] $sanitized_keys
+  *   An array of keys that have been removed.
+  *
+  * @return mixed
+  *   The sanitized input.
+  */
+ protected static function stripDangerousValues($input, array $whitelist, array &$sanitized_keys) {
+   if (is_array($input)) {
+     foreach ($input as $key => $value) {
+       if ($key !== '' && $key[0] === '#' && !in_array($key, $whitelist, TRUE)) {
+         unset($input[$key]);
+         $sanitized_keys[] = $key;
+       }
+       else {
+         $input[$key] = static::stripDangerousValues($input[$key], $whitelist, $sanitized_keys);
+       }
+     }
+   }
+   return $input;
+ }
+
+}
```

تابع "sanitize" پارامترهای فرستاده شده از طریق متدهایGET، POST و کوکی را قبول می‌کند و درخواست را با یک نسخه پاکسازی شده و به دست آمده از تابع stripDangerousValues جایگزین می کند.

'stripDangerousValues'در خط ۸۴ تابع پاکسازی می باشد بطوریکه این تابع بررسی می‌کند که آیا ورودی یک آرایه است و نام کلید هر پارامتر با # شروع می‌شود یا خیر و مقادیر را از نظر آسیب پذیر بودن، پاکسازی می‌نماید.

```
84  + protected static function stripDangerousValues($input, array $whitelist, array &$sanitized_keys) {
85  +   if (is_array($input)) {
86  +     foreach ($input as $key => $value) {
87  +       if ($key !== '' && $key[0] === '#' && !in_array($key, $whitelist, TRUE)) {
88  +         unset($input[$key]);
89  +         $sanitized_keys[] = $key;
90  +       }
91  +       else {
92  +         $input[$key] = static::stripDangerousValues($input[$key], $whitelist, $sanitized_keys);
93  +       }
94  +     }
95  +   }
96  +   return $input;
97  + }
```

مشکل اساسی این است که هسته دروپال (بسیار شبیه سایر فریمورک‌های دیگر) پارامترهای درخواست را به عنوان اشیاء آرایه قبول می‌کند. یک کاربر می‌تواند یک شیء آرایه را به برنامه منتقل کرده و کلیدی که حاوی PayLoad می‌باشد را به عنوان ورودی ارسال کند و دروپال بدون بررسی و پاکسازی، آن را پردازش می‌کند.

۴ منابع

https://www.drupal.org/sa-core-2018-002
https://www.securityfocus.com/bid/103534
https://blog.appsecco.com/remote-code-execution-with-drupal-core-sa-core-2018-002-95e6ecc0c714