PHÂN TÍCH LỖI 1DAY – Joomla

1. **CVE 2019-18650:**
   - Tên lỗi: Cross Site Request Forgery to Remote Code Execution.
   - Phiên bản mắc lỗi: 3.2.0-3.9.12.
   - Điều kiện khai thác: File mã độc đã được upload ở một nơi nào đó ở máy chủ, biết được WEBROOT, admin click vào đường dẫn chứa mã khai thác.
   - Chi tiết lỗi:

   Joomla sử dụng JSession::checkToken đề phòng chống lỗi CSRF nhưng một số function lại không sử dụng JSession::checkToken kiểm tra token người dùng. Từ đó cho phép người dùng gửi input vào không cần đúng token được joomla gen ra.

   Phần code không check token ở đây là ở component com_template. Ở component này cho phép người quản trị thay đổi trực tiếp như thêm, sửa, xóa code ở các template. Ngoài ra các template này được đặt trong WEBROOT. Đa số các function tại component này đều không check lại token sau khi nhận input phía người dùng. Tập trung vào function overrides vì từ function này có thể dẫn tới RCE.

```php
public function overrides()
{
    $app      = JFactory::getApplication();
    $model    = $this->getModel();
    $file     = $app->input->get('file');
    $override = base64_decode($app->input->get('folder'));
    $id       = $app->input->get('id');

    if ($model->createOverride($override))
    {
        $this->setMessage(JText::_('COM_TEMPLATES_OVERRIDE_SUCCESS'));
    }

    // Redirect back to the edit screen.
    $url = 'index.php?option=com_templates&view=template&id=' . $id . '&file=' . $file;
    $this->setRedirect(JRoute::_($url, false));
}
```

Function overrides

Ở controller của template, hàm template này nhận vào 3 param get:

- $file: file cần ghi
- $override: được lưu dưới dạng base64, thư mục chứa file cần ghi.

- $id: id của template (giá trị mặc định).

Và điều quan trọng ở đây là function này nhận các giá trị và không check lại token, dẫn đến bị lỗi CSRF.

Phân tích việc nhận dữ liệu input của Joomla: dựa theo document của Joomla thì các get request sẽ được xử lý theo từng kiểu dữ liệu khác nhau nhằm loại bỏ các lỗi nguy hiểm như Command Injection, XSS, SQL Injection... https://docs.joomla.org/Retrieving_request_data_using_JInput

Ví dụ:

Với input có dạng integer thì Joomla sẽ xử lý như sau:

```
// Only use the first integer value
preg_match('/-?[0-9]+/', (string) $source, $matches);
$result = @ (int) $matches[0];
```

Xử lý integer của joomla

Do dữ liệu là kiểu số nên chỉ nhận các kí tự là số từ 0-9 và chỉ nhận giá trị số đầu tiên. Tiếp theo nếu xử lý path giống như ở function overrides thì Joomla sẽ xử lý như sau.

```
// Converts the input into a string and validates it as a path. (e.g. path/to/file.png or path/to/dir)
// Note: Does NOT accept absolute paths, or paths ending in a trailing slash.
// For a visual representation of the pattern matching used, see http://www.regexper.com/#^[A-Za-z0-9_-]%2B[A-Za-z0-9_\.-]*%28[\\\\\%2F][A-Za-z0-9_-]%2B[A-Za-z0-9_\.-]*%29*%24
// Will return null if the input was invalid.
$pattern = '/^[A-Za-z0-9_-]+[A-Za-z0-9_\.-]*([\\\\\/][A-Za-z0-9_-]+[A-Za-z0-9_\.-]*)*$/';
preg_match($pattern, (string) $source, $matches);
$result = @ (string) $matches[0];
```

Xử lý path của joomla

Cách xử lý path của joomla rất chặt chẽ là không xảy ra lỗi bị Path Traversal hay các lỗi khác tương tự. Tuy nhiên việc xử lý path ở $overrides này sử dụng base64, khi decode lại không kiểm tra sẽ dẫn đến bị lỗi Path Traversal.

```
// Allow a-z, 0-9, slash, plus, equals.
$result = (string) preg_replace('/[^A-Z0-9\/+=]/i', '', $source);
```

Xử lý base64 của joomla

Vậy $overrides bị Path Traversal sẽ được xử lý tiếp tục ở hàm createOverride theo flow của source code. Tiếp tục sau khi decode base64, input không được kiểm tra

để chống lỗi Path Traversal và được đưa vào hàm createTemplateOverride để xử lý.

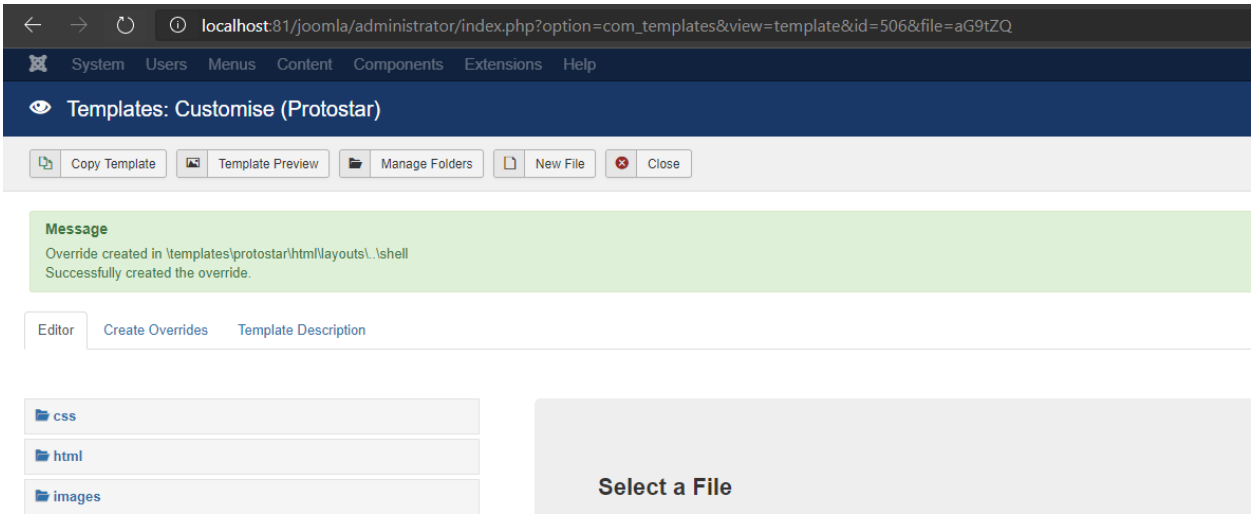
```php
public function createTemplateOverride($overridePath, $htmlPath)
{
    $return = false;

    if (empty($overridePath) || empty($htmlPath))
    {
        return $return;
    }

    // Get list of template folders
    $folders = JFolder::folders($overridePath, null, true, true);

    if (!empty($folders))
    {
        foreach ($folders as $folder)
        {
            $htmlFolder = $htmlPath . str_replace($overridePath, '', $folder);

            if (!JFolder::exists($htmlFolder))
            {
                JFolder::create($htmlFolder);
            }
        }
    }

    // Get list of template files (Only get *.php file for template file)
    $files = JFolder::files($overridePath, '.php', true, true);

    if (empty($files))
    {
        return true;
    }
```

Function createTemplateOverride

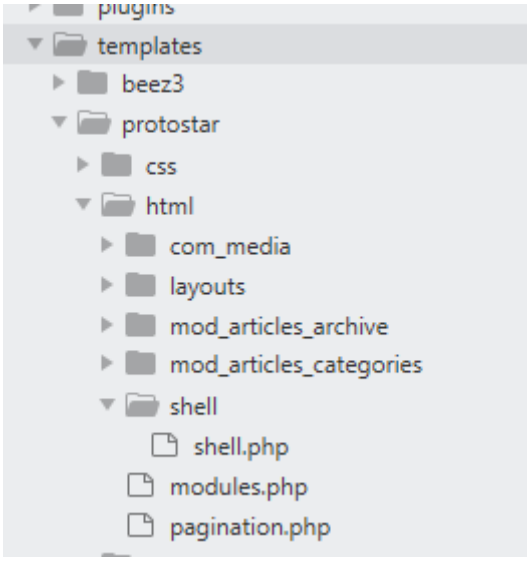
Function sẽ nhận 2 tham số với $overridePath tương ứng với thứ người dùng nhập ban đầu và $htmlPath là thư mục sẽ tương ứng với số id của WEBROOT. Chức năng sẽ lấy toàn bộ file có định dạng đuôi ‘php’ từ thư mục $override để ghi vào $htmlPath. Và vẫn không có kiểm tra phòng chống lỗi Path Traversal. Và từ 2 lỗi CSRF và Path Traversal này ta có được chain attack như sau:

- Tìm cách upload file mã độc lên máy chủ.
- Tìm WEBROOT.
- Lừa admin click vào liên kết có chứa mã độc.

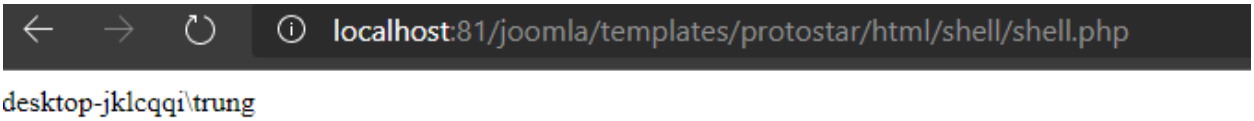
Payload:
http://victim/joomla_path/administrator/index.php?option=com_templates&view=template&task=template.overrides&folder=base64_encode(path_chứa_shell)&id=506&file=aG9tZQ



Sau khi admin bị lừa click vào đường dẫn có chứa mã độc



Đường dẫn chứa file mã độc



Truy cập vào file mã độc

- Cách khắc phục:
  - Kiểm tra đường dẫn sau khi decode base64.
  - Disable các hàm executes command.