GO-TO

CVE-2024-22144

Code Injection

Week 3

Author : Ali Soltani

سلام خوش اومیدن به سومین هفته از GO-TO CVE این هفته به بررسی آسیب‌پذیری پلاگین GOTMLS Plugin که از جمله پلاگین‌های WordPress است، می‌پردازیم. این آسیب‌پذیری با CVSS Score: 9 یک Code Injection است که روی نسخه‌های 4.21.96 => از این پلاگین وجود دارد که بیش از 200000 نصب فعال دارد و در ادامه به بررسی کامل این آسیب‌پذیری می‌پردازیم.

درباره WordPress

وردپرس یک سیستم مدیریت محتوا (CMS)است که به کاربران امکان ساخت و مدیریت وب‌سایت‌ها و وبلاگ‌ها را با سهولت و بدون نیاز به دانش برنامه‌نویسی می‌دهد. این پلتفرم متن‌باز و رایگان است و به زبان PHP نوشته شده و از یک پایگاه داده MySQL برای ذخیره اطلاعات استفاده می‌کند. وردپرس از طریق یک پنل مدیریت کاربرپسند، امکانات گسترده‌ای را برای ایجاد و ویرایش محتوا، مدیریت رسانه‌ها، و تنظیمات وب‌سایت فراهم می‌آورد.

یکی از ویژگی‌های برجسته وردپرس، قابلیت نصب پلاگین‌ها (افزونه‌ها) است. پلاگین‌ها به کاربر این امکان را می‌دهند که بدون نیاز به تغییر کدهای اصلی سیستم، عملکردها و قابلیت‌های جدیدی را به سایت خود اضافه کند. این افزونه‌ها می‌توانند شامل فرم‌های تماس، گالری تصاویر، فروشگاه‌های آنلاین، و بسیاری از امکانات دیگر باشند. نصب پلاگین‌ها به سادگی از طریق مخزن رسمی وردپرس یا آپلود مستقیم فایل‌های پلاگین امکان‌پذیر است و نیاز به تنظیمات پیچیده‌ای ندارد. از طریق این انعطاف‌پذیری، وردپرس توانسته است به یکی از محبوب‌ترین و پرکاربردترین سیستم‌های مدیریت محتوا در سراسر جهان تبدیل شود.

درباره آسیب پذیری

این آسیب‌پذیری‌ها نشان دهنده اهمیت بالای امنیت در طراحی و توسعه پلاگین‌های وردپرس است. توسعه‌دهندگان می‌بایسیت توابع از اندپوینت‌های API را با مکانیزم‌های مبنی بر احراز هویت و اعتبارسنجی مناسب امن کنند و استفاده از توابع امنیتی استاندارد وردپرس، مانند ایجاد و اعتبارسنجی nonce، به جای ایجاد توابع امنیتی سفارشی، توصیه می‌شود تا از بهره‌برداری جلوگیری شود. این امر نه تنها به بهبود امنیت پلاگین کمک می‌کند بلکه از بهره‌برداری‌های احتمالی توسط مهاجمان نیز جلوگیری می‌نماید. در ادامه به بررسی عملکرد مهاجمان در برابر افزونه آسیب‌پذیر می‌پردازیم.

شرحی برآسیب‌پذیری

در حالتی که یک پلاگین به توابع خود از طریق API دسترسی بدون نیاز به احراز هویت دارد، این امر به خطری جدی برای امنیت سیستم تبدیل می‌شود. این به این معناست که هر فردی می‌تواند به این توابع دسترسی پیدا کرده و عملیاتی که ممکن است موجب نقض امنیت سایت شود، را انجام دهد، بدون نیاز به اطلاعات هویتی و مجوزهای معتبر.

در پلاگین GOTMLS، این مسئله به شدت مشهود است. برخی از توابع حساس این پلاگین، از طریق فایل admin-ajax.php بدون نیاز به احراز هویت در دسترس قرار می‌گیرند. این اقدام از طریق افزودن اقدامات با پیشوند wp_ajax_ و nopriv انجام می‌شود، که به این معناست که هر فردی امکان دسترسی به این توابع حساس را دارد، بدون احراز هویت، تا حدی که با یک شروع با حرف «ا»، بتواند این امر را انجام دهد.

این وضعیت، توابعی از قبیل logintime، log_session، load_update و lognewkey را در دسترس قرار می‌دهد.

این توابع ممکن است اطلاعات مهمی را از سرور فاش کنند و حتی امکان دستکاری این اطلاعات را به مهاجمان بدهند، بدون نیاز به احراز هویت یا درخواست مجوز.

```php
<?php

$ajax_functions = array('load_update', 'log_session', 'empty_trash', 'fix', 'logintime', 'lognewkey', 'position', 'scan', 'View_Quarantine', 'whitelist');
// ...
foreach ($ajax_functions as $ajax_function) {
    add_action("wp_ajax_GOTMLS_$ajax_function", "GOTMLS_ajax_nopriv");
    add_action("wp_ajax_nopriv_GOTMLS_$ajax_function", substr($ajax_function, 0, 1) == "l"?"GOTMLS_ajax_$ajax_function":"GOTMLS_ajax_nopriv");
}

?>

```

این قسمت از کد PHP برای اضافه کردن عملکردهای مختلف به AJAX در وردپرس استفاده می‌شود. در اینجا، آرایه‌ای به نام $ajax_functions تعریف شده است که شامل نام توابعی می‌شود که به صورت غیر احراز هویتی قابل دسترس هستند، از جمله load_update, log_session, empty_trash و ... .
سپس با استفاده از یک حلقه foreach، هر عملکرد به نام $ajax_function از آرایه $ajax_functions دریافت شده و به عنوان یک زنجیره از دو اقدام به wp_ajax و wp_ajax_nopriv اضافه می‌شود. این اقدامات امکان دسترسی به توابع مربوطه را برای هر دو نوع کاربر، یعنی هم با و هم بدون احراز هویت، فراهم می‌کنند.
در خصوص توابع مختلف، توابعی مانند logintime اطلاعات server›s microtime را به کاربران بدون نیاز به احراز هویت ارائه می‌دهند. این اطلاعات می‌توانند برای محاسبه و تلاش brute force یک nonce معتبر استفاده شوند.
در این قسمت از کد، مشکلات جدی در ایجاد و اعتبارسنجی Nonce وجود دارد. Nonce به عنوان یک رشته یا عدد تصادفی استفاده می‌شود که هر بار که یک عملیات امنیتی انجام می‌شود، تغییر می‌کند و از این طریق حملات تکراری را کنترل می‌کند.
اما در اینجا، برای ایجاد Nonce از یک الگوریتم ضعیف و ناامن استفاده شده است. از جمله، استفاده از تابع md5 برای تولید Nonce که باعث ایجاد یک رشته یا عدد به نظر تصادفی می‌شود، اما در واقع از ورودی‌های قابل پیش‌بینی یا شناخته شده استفاده می‌کند. به عبارت دیگر، ورودی‌های قابل پیش‌بینی خروجی‌های پیش‌بینی‌پذیری را ایجاد می‌کنند. این باعث افزایش آسیب‌پذیری سیستم می‌شود و به افراد بد جایزی نیازی برای حدس زدن یا انجام حملات brute force ندارند تا این مقادیر را به صورت مستقیم حدس بزنند یا انجام دهند.
کد مورد اشاره به صورت زیر است:

```php
md5(substr(number_format(microtime(true), 9, '-', '/'), 6).GOTMLS_installation_key.GOTMLS_plugin_path);
```

این کد از تابع microtime برای دریافت زمان کنونی سرور استفاده می‌کند، سپس از تابع substr برای استخراج یک بخش معین از این زمان استفاده می‌کند. سپس این بخش زمان به کلید نصب و مسیر پلاگین installation_key و pl- ugin_path افزوده می‌شود و در نهایت با استفاده از تابع md5 به عنوان Nonce تولید می‌شود.

این روش غیرامن و ضعیف است، زیرا از ورودی‌هایی استفاده می‌کند که در بسیاری از موارد قابل پیش‌بینی هستند، ممکن است نتایج تصادفی تولید نشود و به این ترتیب سیستم قابل حمله‌تر خواهد شد.

در این بخش از توضیحات، به مسائل امنیتی مربوط به تولید و اعتبارسنجی Nonce در پلاگین مورد بحث می‌پردازیم. Nonce به عنوان یک رشته یا عدد تصادفی استفاده می‌شود که هر بار که یک عملیات امنیتی انجام می‌شود، تغییر می‌کند و از این طریق حملات تکراری را کنترل می‌کند.

در اینجا، Nonce به شکل غیر ایمنی تولید و اعتبارسنجی می‌شود. به عنوان مثال، مسیر پلاگین GOTMLS_plugin_ path قابل پیش‌بینی است و کلید نصب GOTMLS_installation_key نیز از طریق آدرس URL سایت قابل شناسایی است. همچنین، سرور زمان کنونی خود را با دقت تا 4 رقم اعشار منتشر می‌کند. با توجه به این موارد، حتی با دقت زمان دقیق می‌توان از روش brute force برای حدس زدن Nonce معتبر استفاده کرد؛ اما این عملیات نیازمند تست میلیون‌ها تابع هش است که زمان زیادی را می‌طلبد. برای بهبود دقت در تخمین زمان ایجاد Nonce، می‌توان با بررسی زمان کنونی سرور قبل و بعد از تولید Nonce، به کمک محاسبات ریاضی، زمان ایجاد را به طور دقیق‌تر پیش‌بینی کرد.

یکی از جنبه‌های کلیدی امنیت Nonce این است که آنها به جلسات خاص مرتبط می‌شوند. این عملیات امنیتی اضافی، هر Nonce را با یک جلسه کاربری یکتا مرتبط می‌کند، که باعث می‌شود برای حملات‌کنندگان بسیار سخت‌تر باشد تا از آنها بهره ببرند. اما در این پلاگین، Nonceها به صورت گلوبال ذخیره می‌شوند و بین کاربران مختلف استفاده می‌شوند.

در این پلاگین، nonce به صورت گلوبال ذخیره شده و بین کاربران با سطح دسترسی مختلف باز استفاده می‌شوند. همچنین تابع اعتبار سنجیدnonce به صورت زیر است:

```
if (is_array($_REQUEST["GOTMLS_mt"])) {
    foreach ($_REQUEST["GOTMLS_mt"] as $_REQUEST_GOTMLS_mt)
        if (strlen($_REQUEST_GOTMLS_mt) == 32 && isset($GLOBALS["GOTMLS"]["tmp"]["nonce"][$_REQUEST_GOTMLS_mt]))
            return (INT) $GLOBALS["GOTMLS"]["tmp"]["nonce"][$_REQUEST_GOTMLS_mt];
    return 0;
}
elseif (strlen($_REQUEST["GOTMLS_mt"]) == 32 && isset($GLOBALS["GOTMLS"]["tmp"]["nonce"][$_REQUEST["GOTMLS_mt"]])) {
    return (INT) $GLOBALS["GOTMLS"]["tmp"]["nonce"][$_REQUEST["GOTMLS_mt"]];
}
```

این روش، تعداد درخواست‌های لازم برای brute force یک nonce معتبر را به طور قابل توجهی کاهش می‌دهد زیرا ما می‌توانیم چندین حدس را در یک درخواست ارسال کنیم. تعداد متغیرهایی که می‌توان در یک درخواست HTTP ارسال کرد فقط با max_input_vars در PHP محدود می‌شود که معمولاً به 1000 محدود است.

یک نقص دیگر این است که مهاجم می‌تواند تمامیnonces های موجود در سرور را نامعتبر کند و تا 25 عدد را با هر درخواست ایجاد کند. که این موضوع به مهاجم امکان می‌دهد تا درمدت زمان کمتری بروت فورس خود را انجام دهد.

درادامه به نوشتن قطعه کدی می‌پردازیم که بتواند به نوعی بازسازی این حمله باشد که نتیجه آن را در تصویر زیر مشاهده می‌فرمایید:

```
Current microtime: 1716462518.6453
Input string: 2518.6453124b65e39d7b19df6a45cdaa16dc6044/var/www/html/wp-content/plugins/gotmls/
Generated nonce (MD5 hash): f10739180cf31f13da4b6704ead6ee35
Current microtime: 1716462518.6463
Input string: 2518.6463124b65e39d7b19df6a45cdaa16dc6044/var/www/html/wp-content/plugins/gotmls/
Generated nonce (MD5 hash): e6673e566ae581932ff9ab0d583e0969
Current microtime: 1716462518.6463
Input string: 2518.6463124b65e39d7b19df6a45cdaa16dc6044/var/www/html/wp-content/plugins/gotmls/
Generated nonce (MD5 hash): e6673e566ae581932ff9ab0d583e0969
f10739180cf31f13da4b6704ead6ee35
e6673e566ae581932ff9ab0d583e0969
80c0ca34c15d604759b6ad22f98cff9b
a2f8b92fc62ad938ff3d0210eb51b14a
Nonces after clearing:
{}
```

```python
import hashlib
import time

def generate_nonce(installation_key: str, plugin_path: str) -> str:

    current_microtime = str(round(time.time(), 4))

    print(f"Current microtime: {current_microtime}")

    input_string = current_microtime[6:] + installation_key + plugin_path

    print(f"Input string: {input_string}")

    hash_result = hashlib.md5(input_string.encode()).hexdigest()

    print(f"Generated nonce (MD5 hash): {hash_result}")

    return hash_result

def validate_nonce(nonce: str, stored_nonces: dict) -> int:

    if nonce in stored_nonces:
        return int(stored_nonces[nonce])
    else:
        return 0

def clear_nonces(stored_nonces: dict):

    stored_nonces.clear()

def exploit_vulnerability(installation_key: str, plugin_path: str, stored_nonces: dict):
    for _ in range(25):
        generated_nonce = generate_nonce(installation_key, plugin_path)
        stored_nonces[generated_nonce] = 12345

installation_key = "124b65e39d7b19df6a45cdaa16dc6044"
plugin_path = "/var/www/html/wp-content/plugins/gotmls/"

stored_nonces = {}

exploit_vulnerability(installation_key, plugin_path, stored_nonces)

print("Stored nonces:")
for nonce in stored_nonces:
    print(nonce)

clear_nonces(stored_nonces)

print("Nonces after clearing:")
print(stored_nonces)
```

این کد به زبان پایتون چندین تابع مختلف برای تولید و مدیریت nonceها (شماره‌های یکبار مصرف) ایجاد می‌کند. در ادامه به تشریح هر بخش از این کد می‌پردازم.

1. تابع generate_nonce
این تابع یک nonce جدید را بر اساس ورودی‌های installation_key و plugin_path تولید می‌کند.

ورودی‌ها:
- یک رشته که به عنوان کلید نصب استفاده می‌شود: installation_key
- مسیر افزونه : plugin_path

خروجی:
- یک رشته که همان nonce تولید شده است.

مراحل:
- دریافت زمان میکروثانیه جاری: زمان جاری سیستم را تا 4 رقم اعشار گرفته و به رشته تبدیل می‌کند.
((current_microtime = str(round(time.time(), 4

- ساخت رشته ورودی: یک رشته جدید با ترکیب بخش اعشاری زمان جاری، کلید نصب و مسیر افزونه ساخته می‌شود.
input_string = current_microtime[6:] + installation_key + plugin_path

- محاسبه هش: MD5 هش MD5 از رشته ورودی محاسبه و نتیجه برگردانده می‌شود.
hash_result = hashlib.md5(input_string.encode()).hexdigest()

2.تابع validate_nonce
این تابع بررسی می‌کند که آیا یک nonce خاص در مجموعه nonces ذخیره شده وجود دارد یا خیر.

ورودی‌ها:
- برای بررسی nonce : nonce
- های ذخیره شده‌nonce دیکشنری شامل : stored_nonces

خروجی:
- عدد صحیح که اگر nonce موجود باشد، مقدار ذخیره شده برمی‌گردد و در غیر این صورت 0.

مراحل:
- بررسی وجود nonce :

```
if nonce in stored_nonces:
    return int(stored_nonces[nonce])
else:
    return 0
```

3. تابع clear_nonces
این تابع تمامی nonceهای ذخیره شده را پاک می‌کند.

ورودی‌ها:
- stored_nonces : دیکشنری شامل nonceهای ذخیره شده

مراحل:
- پاک‌سازی دیکشنری:

```
stored_nonces.clear()
```

4. تابع exploit_vulnerability
این تابع به منظور تولید و ذخیره تعداد زیادی nonce استفاده می‌شود.

ورودی‌ها:
- installation_key : کلید نصب
- plugin_path : مسیر افزونه
- stored_nonces : دیکشنری شامل nonceهای ذخیره شده

مراحل:
- تولید و ذخیره nonce در حلقه: 25 nonce تولید و در دیکشنری ذخیره می‌کند.

```
for _ in range(25):
    generated_nonce = generate_nonce(installation_key, plugin_path)
    stored_nonces[generated_nonce] = 12345
```

مثال کاربرد

در انتها، مثالی از چگونگی استفاده از این توابع ارائه شده است.
- کلید نصب و مسیر افزونه:

```
installation_key = «your-installation_key «
plugin_path = «your-path»
```

- **ذخیره nonceها :**

```
stored_nonces = {}
exploit_vulnerability(installation_key, plugin_path, stored_nonces)
```

- **چاپ nonceهای ذخیره شده:**

```
for nonce in stored_nonces:
    print(nonce)
```

- **پاک‌سازی nonceها:**

```
clear_nonces(stored_nonces)
```

نتیجه نهایی تحلیل کد :
این کد نشان می‌دهد که چگونه می‌توان nonce تولید کرد، آنها را ذخیره کرد، صحت آنها را بررسی کرد و در صورت نیاز، تمامی آنها را پاک کرد. این قابلیت‌ها می‌تواند در زمینه‌هایی پیدا کردن آسیب پذیری به پنتستر ها کمک شایانی کند .

بهره برداری ازآسیب پذیری

- **اجرای کد از راه دور با تزریق قوانین بدافزار مخرب**

این حمله از طراحی و پیاده‌سازی ناامن توابع API مدیریتی یک پلاگین بهره‌برداری می‌کند تا به اجرای کد از راه دور (RCE) دست یابد. در ادامه توضیحی کامل بر مراحل این حمله آورده شده است.

1. پیدا کردن مقدار نانس از طریق بروت فورس

2. دسترسی به توابع AJAX:
پس از به دست آوردن نانس، مهاجم به تمامی توابع AJAX، از جمله توابع مدیریتی که نباید عمومی باشند، دسترسی پیدا می‌کند. این شامل توابعی است که با ‹ا› شروع می‌شوند و دیگر توابع خصوصی.

3. تابع GOTMLS_ajax_load_update
این تابع کلیدی است. این تابع اجازه به‌روزرسانی تعاریف بدافزار کدگذاری شده‌ی base64 و سریال شده‌ی PHP را می‌دهد. این تابع باید پرایوت باشد اما به دلیل بروت فورس نانس قابل دسترسی است.

4. سریال‌سازی ناامن:
با اینکه تابع دارای یک محافظت کوچک regex در برابر حملات سریال‌سازی ناامن است، این محافظت کافی نیست. مهاجم همچنان می‌تواند الگوهای regex سفارشی را به تعاریف بدافزار تزریق کند.

5. ساخت الگوهای مخرب : regex

مهاجم الگوهای خاص regex را برای دستکاری کد منبع پلاگین ایجاد می‌کند. هدف حذف انتخابی کد و باقی گذاشتن یک قطعه کد قابل بهره‌برداری است. در این مثال، سه الگوی regex استفاده شده میپردازیم :

```
$inj = array("known" => array(
            "stealthcopter testing 1" => array(0 => 'M4t01', 1 => '/\$bad = array\("/'),
            "stealthcopter testing 2" => array(0 => 'M4t02', 1 => '/", "preg_replace.*?isset/s'),
            "stealthcopter testing 3" => array(0 => 'M4t03', 1 => '/&&is_numeric\(.*?\n\)(?=;)/s')
        ));
```

6. به‌روزرسانی تعاریف بدافزار:

تعاریف مخرب سریال و کدگذاری شده به تابع AJAX load_update ارسال می‌شوند. تعاریف بدافزار با این الگوهای regex مخرب به‌روزرسانی می‌شوند.

7. اجرای اسکن برای بهره برداری از آسیب پذیری:

مهاجم یک اسکن را روی فایل مشخصیمثلاً images/index.php آغاز می‌کند. اسکن با استفاده از الگوهای regex مخرب فایل را اصلاح می‌کند.حذف کد باعث می‌شود یک قطعه کد آسیب‌پذیر باقی بماند مانند

eval($_REQUEST[«mt»]);

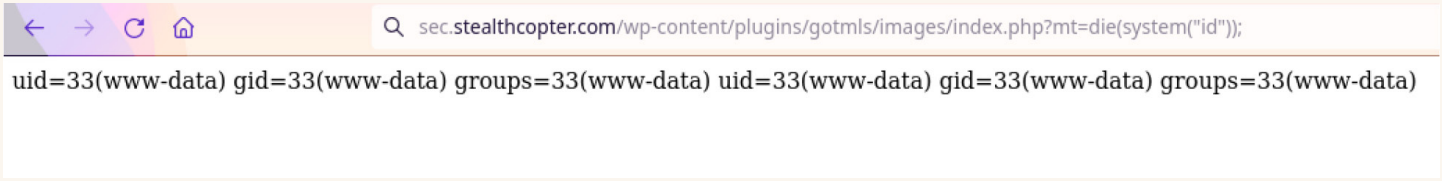
این قطعه کد اجازه اجرای کد PHP دلخواه را می‌دهد. مهاجم اکنون می‌تواند هر کد PHP را از طریق پارامتر mt ارسال کند تا روی سرور اجرا شود.

8. اجرای دستورات:

به عنوان مثال، برای اجرای دستورات سیستمی، مهاجم می‌تواند ارسال کند:

die(system(«id»));

این دستور توسط سرور اجرا می‌شود و مهاجم کنترل کامل را به دست می‌آورد.



خلاصه نهایی

این حمله مجموعه‌ای از مراحل را نشان می‌دهد که با ترکیب بروت فورس نانس، بهره‌برداری از دسترسی ناامن به توابع AJAX، و دستکاری داده‌های سریال شده PHP برای تزریق الگوهای regex مخرب به اجرای کد از راه دور منجر می‌شود.

نتیجه نهایی ایجاد درب پشتی است که به مهاجم اجازه می‌دهد کد PHP دلخواه را روی سرور اجرا کند و امنیت آن را کاملاً به خطر بیاندازد

جلوگیری

برای جلوگیری از این آسیب پذیری کافی است فقط نسخه پلاگین خود را ارتقادهدید وهمیشه به توسیه های امنیتی مشاروان امنیت خود گوش دهید به امید فردایی امن تر.

منابع

https://patchstack.com/
https://sec.stealthcopter.com/
https://cve.mitre.org/