GO-TO CVE

CVE-2023-49103 & CVE-2023-49105

Information Disclosure & Privilege escalation
Week 5

Author : Ali Soltani

مقدمه

سلام خوش آمدید به هفته ششم از این برنامه. این هفته به بررسی دو تا CVE می‌پردازیم که از ترکیب این دو به RCE می‌رسیم. آسیب‌پذیری اول مربوط به CVE-2023-49103 هست که پیدا کردن PHPinfo هست که شاید تصور کنید شدت خاصی ندارد، ولی دراین سناریو نقش مهمی را بازی می‌کند که CVSS آن را تا 10 ارتقا داده است و CVE-2023-49105 که 9.8CVSS که را به خود اختصاص داده است و ورژن 10.13.1وپایین‌تر را در برمی‌گردد که به بررسی آن می‌پردازیم.

درباره OwnCloud

OwnCloud یک پلتفرمی است که برای ذخیره‌سازی، همگام‌سازی، و به اشتراک‌گذاری فایل‌ها به کار می‌رود. این پلتفرم به کاربران و سازمان‌ها اجازه می‌دهد تا فایل‌های خود را روی سرورهای خصوصی خودشان ذخیره کنند و از هر کجا و با هر دستگاهی به آنها دسترسی داشته باشند. برخی از ویژگی‌ها و مزایای ownCloud عبارتند از:

1. ذخیره‌سازی و همگام‌سازی فایل‌ها: ownCloud به کاربران اجازه می‌دهد تا فایل‌های خود را در سرور خصوصی‌شان ذخیره کنند و با دستگاه‌های مختلف همگام‌سازی کنند. این به معنی دسترسی آسان به فایل‌ها از طریق کامپیوتر، موبایل، و تبلت است.

2. اشتراک‌گذاری فایل‌ها: کاربران می‌توانند فایل‌ها و پوشه‌ها را با دیگران به اشتراک بگذارند، چه درون سازمان و چه با کاربران خارجی.

3. امنیت و کنترل: با استفاده از OwnCloud، داده‌ها روی سرورهایی که توسط کاربران مدیریت می‌شود ذخیره می‌شوند، بنابراین کاربران کنترل کاملی بر روی داده‌های خود دارند. این برخلاف سرویس‌های ذخیره‌سازی ابری عمومی است که داده‌ها روی سرورهای شخص ثالث قرار دارند.
4. گسترش‌پذیری: ownCloud دارای پلاگین‌ها و اپلیکیشن‌های مختلفی است که می‌توانند برای اضافه کردن ویژگی‌ها و قابلیت‌های جدید به سیستم اضافه شوند. این پلاگین‌ها می‌توانند برای ادغام با ابزارهای دیگر یا اضافه کردن امکانات سفارشی‌سازی به کار روند.

5. دسترسی از طریق وب و موبایل: ownCloud از طریق یک رابط وب کاربر پسند و همچنین اپلیکیشن‌های موبایل برای سیستم‌عامل‌های مختلف در دسترس است، که این امکان را فراهم می‌کند تا کاربران از هر کجا به فایل‌های خود دسترسی داشته باشند.
ownCloud به عنوان یک راه‌حل منبع‌باز open source عرضه می‌شود، این پلتفرم به خصوص برای سازمان‌هایی که نیاز به کنترل کامل بر روی داده‌های خود دارند، مانند شرکت‌ها، مؤسسات آموزشی، و سازمان‌های دولتی، مفید است.

CVE-2023-49103

همان‌طور که در مقدمه گفتم این آسیب‌پذیری افشا شدن مسیر PHPinfo است. در واقع، در نصب‌های Docker، اگر کسی بتواند به صفحه PHPinfo دسترسی پیدا کند، به تمام متغیرهای محیطی و اطلاعات حساس مانند نام کاربری و رمز عبور مدیر ownCloud دسترسی پیدا می‌کند.
صفحه PHPinfo به‌طور معمول برای توسعه‌دهندگان و مدیران سیستم استفاده می‌شود تا اطلاعاتی در مورد تنظیمات PHP و محیط سرور بدست آورند. این صفحه شامل اطلاعاتی درباره نسخه PHP، ماژول‌های بارگذاری شده، متغیرهای محیطی، و تنظیمات مختلف PHP است.
مشکل اصلی اینجاست که متغیرهای محیطی ممکن است حاوی اطلاعات بسیار حساسی باشند، مانند نام کاربری و رمز عبور مدیر سیستم، کلیدهای API، و سایر اطلاعات محرمانه. در نصب‌های Docker، اگر صفحه PHPinfo به‌درستی محافظت نشود و دسترسی به آن برای هر کسی ممکن باشد، مهاجمان می‌توانند با دسترسی به این صفحه به این اطلاعات حساس دسترسی پیدا کنند.
برای مثال، در نصب ownCloud، اگر کسی بتواند به صفحه PHPinfo دسترسی پیدا کند، می‌تواند متغیرهای محیطی را ببیند که ممکن است شامل اطلاعات زیر باشند:

- نام کاربری مدیر ownCloud
- رمزعبور مدیر ownCloud

این اطلاعات می‌توانند به مهاجمان اجازه دهند تا به سیستم ownCloud دسترسی کامل پیدا کنند، تنظیمات را تغییر دهند، فایل‌ها را دستکاری کنند، و حتی کدهای مخرب را اجرا کنند.
برای جلوگیری از چنین حملاتی، باید مطمئن شد که دسترسی به صفحه PHPinfo محدود به افراد مجاز است و این صفحه برای عموم قابل دسترسی نباشد. همچنین، استفاده از روش‌های امنیتی مانند احراز هویت قوی و تنظیمات مناسب فایروال می‌تواند از دسترسی غیرمجاز به این صفحه جلوگیری کند.

CVE-2023-49103

در این آسیب‌پذیری فرد مهاجم بدون داشتن حساب کاربری می‌توانند کنترل کامل CRUD هر فایل از هر حساب کاربری را به دست آورند. در برخی موارد، آن‌ها ممکن است قادر به اجرای کد از راه دور RCE باشند. مهاجمانی که اطلاعات ورود یک حساب را دارند، می‌توانند به سطح ادیمن ارتقا یابند و RCE برسند.

از کاربر تا RCE:

هنگام ارسال درخواست‌ها به برخی بخش‌های وبسایت، از جمله WEBDAV و CALDAV، کاربران می‌توانند با ارائه نام کاربری و امضا دیجیتال، احراز هویت کنند. امضا دیجیتال با استفاده از کلید خاص کاربر و عناصر موجود در درخواست HTTP، مانند پارامترهای GET، متد HTTP و غیره دریافت و محاسبه می‌شود. متأسفانه، به طور پیش‌فرض، برای کاربران کلیدی تنظیم نکرده‌اند. در این حالت، کلید امضای آن‌ها به صورت رشته‌ای خالی تعریف شده است. در نتیجه، یک مهاجم غیرمجاز می‌تواند هر کاربری را با دانستن نام کاربری آن‌ها جعل هویت کند.

دسترسی به WEBDAV به عنوان یک مهاجم پتانسیل زیادی دارد: می‌توان هر فایل کاربر دسترسی خواندن و نوشتن داشته باشد به هرفایلی درون سرور. هرگاه فایل‌هایی از نوع خاص (مثلاً تصاویر) آپلود می‌شوند، پیش‌نمایشی ایجاد می‌شود که برای فرمت‌های خاص فایل، own-Cloud از ImageMagick برای تولید این پیش‌نمایش استفاده می‌کند. اگر این کتابخانه به‌روز نباشد، مهاجم می‌تواند RCE برسد. با این حال، اگر ImageMagick آسیب‌پذیر نباشد و شما از قبل حساب کاربری داشته باشید، راه دیگری برای به دست آوردن RCE وجود دارد که آن ارتقای سطح دسترسی خود به ادیمن است. درادامه به بررسی کد احراز هویت ادمین می‌پردازیم.

```php
# /apps/dav/lib/Connector/Sabre/Auth.php

$verifier = new Verifier($request, $this->config);
if ($verifier->isSignedRequest()) {
    if (!$verifier->signedRequestIsValid()) {
        return [false, 'Invalid url signature'];
    }
    // TODO: setup session ???
    $urlCredential = $verifier->getUrlCredential();
    $user = \OC::$server->getUserManager()->get($urlCredential);
    if ($user === null) {
        $message = \OC::$server->getL10N('dav')->t('User unknown');
        throw new LoginException($message);
    }
    if (!$user->isEnabled()) {
        $message = \OC::$server->getL10N('dav')->t('User disabled');
        throw new LoginException($message);
    }
    $this->userSession->setUser($user); // <--- here
    \OC_Util::setupFS($urlCredential);
    $this->session->close();
    return [true, $this->principalPrefix . $urlCredential];
}
```

توضیح کد بالا به شرح زیر است:

این کد وظیفه احراز هویت درخواست‌ها از طریق URL را بر عهده دارد. بیایید به هر قسمت کد به صورت جداگانه نگاه کنیم:

$verifier = new Verifier($request, $this->config);

یک شیء از کلاسVerifier ایجاد می‌شود که مسئول بررسی درخواست‌ها است.Verifier با استفاده از پارامترهای درخواستrequest و

تنظیمات پیکربندی ساخته می‌شود.

```
if ($verifier->isSignedRequest())
```

چک می‌کند که آیا درخواست امضا شده است یا خیر.

```
{ return [false, ‹Invalid url signature›];
```

شرطی دارد که اگر درخواست امضا شده معتبر نباشد، یک پیغام خطا بازگشت داده می‌شود و روند ادامه نمی‌یابد.

```
urlCredential = $verifier->getUrlCredential();$
```

در صورت معتبر بودن امضا، اعتبارنامه‌های مربوط به URL از کردنشیال را دریافت می‌کند.

```
user = \OC::$server->getUserManager()->get($urlCredential);$
```

با استفاده از اعتبارنامه‌ها، تلاش می‌شود کاربر مربوطه از طریق مدیریت کاربرUserManager بارگذاری شود.

```
if ($user === null) {
    message = \OC::$server->getL10N(‹dav›)->t(‹User unknown›);$
    throw new LoginException($message);
  }
```

اگر کاربر پیدا نشود، یک استثنا با پیغام «کاربر ناشناس» آلرت می‌شود.

```
  if (!$user->isEnabled()) {
message = \OC::$server->getL10N(‹dav›)->t(‹User disabled›);$
    throw new LoginException($message);
```

اگر کاربر غیرفعال باشد، یک استثنا با پیغام «کاربر غیرفعال» آلرت می‌شود.

```
this->userSession->setUser($user); / <--- here$
```

کاربر معتبر به نشست کاربری فعلی تنظیم می‌شود. این نقطه‌ای است که کاربر به سیستم وارد می‌شود.

```
  OC_Util::setupFS($urlCredential);\
```

سیستم فایل کاربر با استفاده از اعتبارنامه‌های URL تنظیم می‌شود.

```
this->session->close();$
  return [true, $this->principalPrefix . $urlCredential];
```

نشست بسته می‌شود و یک پاسخ موفقیت‌آمیز همراه با اطلاعات کاربری برگردانده می‌شود.
به طور کلی، این کد تلاش می‌کند تا درخواست‌های امضا شده را احراز هویت کند، کاربر مربوطه را بازیابی کند، و در صورت موفقیت، نشست کاربری را تنظیم و سیستم فایل کاربر را آماده کند. درنهایت، یک پاسخ موفقیت‌آمیز ارسال می‌شود. متغیر نشست user_id تنها متغیری است که هنگام احراز هویت با استفاده از یک URL امضا شده تنظیم می‌شود. این متغیر نام کاربر وارد شده را ذخیره می‌کند. با این روش، سیستم می‌تواند تشخیص دهد که کدام کاربر در حال حاضر وارد شده و عملیات‌های مرتبط با آن کاربر را مدیریت کند. تنظیم این متغیر همچنین به معنی فعال‌سازی کاربر در نشست است که امکان دسترسی به منابع و اطلاعات مرتبط با آن کاربر را فراهم می‌کند.
حالا اگر بخواهیم که دسترسی داشته باشیم به صفحات آن کاربر مثل تنظیمات حساب کاربری نحوه انجام آن به صورت زیر است که روی کد توضیح داده می‌شود.

```
if (\OC::$server->getUserSession()) {
request = \OC::$server->getRequest();$
session = \OC::$server->getUserSession();$
davUser = \OC::$server->getUserSession()->getSession()->get(‹AUTHENTICATED_TO_DAV_BACKEND›);$
  if ($davUser === null) {
session->validateSession();$
```

1.بررسی وجود نشست کاربر:

```
  if (\OC::$server->getUserSession()) {
```

این شرط بررسی می‌کند که آیا نشست کاربری UserSession وجود دارد یا نه.
2. دریافت درخواست و نشست:

```
request = \OC::$server->getRequest();$
session = \OC::$server->getUserSession();$
```

درخواست فعلی و نشست کاربر دریافت می‌شود.
3. بررسی AUTHENTICATED_TO_DAV_BACKEND

```
davUser = \OC::$server->getUserSession()->getSession()->get(‹AUTHENTICATED_TO_DAV_BACKEND›);$
```

متغیر AUTHENTICATED_TO_DAV_BACKEND از نشست کاربر استخراج می‌شود. این متغیر نشان‌دهنده این است که آیا کاربر به صورت معتبر به پشت‌صحنه DAV (Distributed Authoring and Versioning) احراز هویت شده است یا خیر.

```
  if ($davUser === null) {
session->validateSession();$
```

اگر AUTHENTICATED_TO_DAV_BACKEND تهی باشد، تابع validateSession() فراخوانی می‌شود. این تابع بررسی می‌کند که آیا نشست یک توکن معتبر دارد یا خیر.
تابع validateSession مسئول بررسی اعتبار نشست است. اگر توکن معتبری وجود نداشته باشد، کاربر به صورت خودکار از سیستم خارج می‌شود.

نتیجه‌گیری

از آنجایی که در این مورد، AUTHENTICATED_TO_DAV_BACKEND تنظیم نشده است، تابع validateSession فراخوانی می‌شود که بررسی می‌کند نشست کاربر دارای توکن معتبر است یا خیر. چون هیچ توکنی تنظیم نشده است، کاربر از سیستم خارج می‌شود و نمی‌تواند به صفحات استاندارد وبسایت دسترسی پیدا کند بنابراین، این باگ به تنهایی کافی نیست تا به صفحات استاندارد دسترسی پیدا کنید، زیرا سیستم احراز هویت اعتبار نشست را بررسی می‌کند و در صورت نبودن توکن معتبر، کاربر را از سیستم خارج می‌کند.
در این بخش، فرآیند احراز هویت به جزئیات بیشتری پرداخته می‌شود، به خصوص با بررسی تابع `verifyAuthHeaders` که اطمینان می‌دهد کاربر به درستی احراز هویت شده است. بیایید به جزئیات کد و نحوه عملکرد آن بپردازیم:
1. فراخوانی متد auth برای هر ماژول احراز هویت:

```
user = $module->auth($request); # [1]$
```

در اینجا، برای هر ماژول احراز هویت، متد auth با استفاده از درخواست request فراخوانی می‌شود تا کاربر احراز هویت شود. اگر ماژول بتواند

کاربر را احراز هویت کند، یک شیء کاربر برگردانده می‌شود.
2. بررسی کاربر احراز هویت شده:

```
  if ($user !== null) {
    if ($this->isLoggedIn() && $this->getUser()->getUID() !== $user->getUID()) { / [2]
shallLogout = true;$
        break;
```

اگر کاربر معتبر باشد سیستم بررسی می‌کند که آیا کاربر هم اکنون وارد شده است و آیا شناسه کاربر فعلی با شناسه کاربر احراز هویت شده مطابقت دارد یا نه اگر شناسه‌ها متفاوت باشند، متغیر shallLogoutبه true تنظیم می‌شود و حلقه متوقف می‌شود.
3. خروج از سیستم در صورت بروز مشکل در نشست:

```
  if ($shallLogout) { # [3]
    / the session is bad -> kill it
    this->logout();$
    return false;
```

اگر shallLogout برابر با true باشد، به این معنی است که نشست نامعتبر است و سیستم از نشست فعلی خارج می‌شود بازگردانده می‌شود. در غیر این صورت، true بازگردانده می‌شود.

نتیجه‌گیری

تابع verifyAuthHeaders بررسی می‌کند که کاربر به درستی احراز هویت شده باشد. اگر هر یک از ماژول‌های احراز هویت کاربر را احراز هویت کنند، سیستم اطمینان حاصل می‌کند که کاربر احراز هویت شده با کاربر ذخیره شده در نشست user_id مطابقت دارد. در صورتی که مطابقت نداشته باشند، کاربر از سیستم خارج می‌شود.
برای دور زدن این بررسی، تنها راه این است که متد auth در ماژول احراز هویت TokenAuthModule مقدار null برگرداند، حتی در صورتی که نشست معتبری وجود داشته باشد. این موضوع تضمین می‌کند که کاربر با استفاده از نشست ذخیره شده، احراز هویت نمی‌شود و بررسی مطابقت شناسه‌ها انجام نمی‌شود.

liangroup.net

```php
class TokenAuthModule implements IAuthModule {
    ...

    public function auth(IRequest $request) {
        ...
        $dbToken = $this->getToken($request, $token); # [1]
        if ($dbToken === null) {
            return null;
        }
        ...
        $uid = $dbToken->getUID();
        return $this->manager->get($uid);
    }

    private function getToken(IRequest $request, &$token) {
        $authHeader = $request->getHeader('Authorization'); # [2]
        if ($authHeader === null || \strpos($authHeader, 'token ') === false)
{
            // No auth header, let's try session id
            try {
                $token = $this->session->getId();
            } catch (SessionNotAvailableException $ex) {
                return null;
            }
        } else {
            $token = \substr($authHeader, 6);
        }

        try {
            return $this->tokenProvider->getToken($token);
        } catch (InvalidTokenException $ex) {
            $token = null;
            return null;
        }
    }
}
```

نحوه افزایش دسترسی با استفاده از اشکال امنیتی:

این توضیح گام به گام نحوه افزایش دسترسی از یک کاربر عادی به حساب مدیر (admin) را شرح می‌دهد، با استفاده از اشکال امنیتی در سیستم احراز هویت. بیایید مراحل را بررسی کنیم.

ما نیاز داریم که متد getToken()مقدار NULL برگرداند تا چک اعتبار کاربر نادیده گرفته شود. خوشبختانه، این متد ابتدا بررسی می‌کند که آیا یک توکن در هدر Authorization وجود دارد یا نه. با ارائه یک توکن نادرست، می‌توانیم متد TokenAuthModule::auth() را مجبور به بازگرداندن مقدار NULL کنیم، حتی اگر کاربر وارد شده باشد. این کار بررسی نام کاربری را دور می‌زند.

مراحل افزایش دسترسی:

1. ورود به سیستم به عنوان یک کاربر عادی:
   - به حساب کاربری استاندارد خود وارد شوید.
   - این کار شما را با یک نشست معتبر به سیستم وارد می‌کند.

2. استفاده از مکانیزم امضای URL برای جعل user_id
   - از مکانیزم امضای URL استفاده کنید تا user_id خود را جعل کنید.
   - این کار به شما امکان می‌دهد به منابع کاربر دیگر دسترسی پیدا کنید.

3. دسترسی به هر صفحه با استفاده از هدر Authorization نادرست:
   - به هر صفحه‌ای دسترسی پیدا کنید، اما در هدر درخواست، یک توکن نادرست مانند Authorization: token thisisnotavalidtoken قرار دهید.
   - این کار باعث می‌شود که متد TokenAuthModule::auth()مقدار NULL برگرداند، حتی اگر شما وارد شده باشید.

بیایید یک مثال عملی از نحوه اجرای این مراحل ارائه دهیم:

```php
// ورود به سیستم به عنوان کاربر عادی
$loginResponse = $client->post('/login', [
    'form_params' => [
        'username' => 'standard_user',
        'password' => 'password123',
    ],
]);

// استخراج کوکی نشست برای درخواست‌های بعدی
$cookies = $loginResponse->getHeader('Set-Cookie');

// استفاده از URL امضا شده برای جعل user_id
$signedUrl = "https://example.com/some_path?user_id=admin&signature=abc123";

// نادرست Authorization ارسال درخواست به صفحه‌ای با هدر
$response = $client->get($signedUrl, [
    'headers' => [
        'Authorization' => 'token thisisnotavalidtoken',
        'Cookie' => $cookies,
    ],
]);

// بررسی دسترسی به صفحه به عنوان کاربر مدیر
if ($response->getStatusCode() === 200) {
    echo "Privilege escalation successful!";
} else {
    echo "Privilege escalation failed.";
}
```

نتیجه‌گیری

این فرآیند به شما امکان می‌دهد تا از یک کاربر عادی به حساب مدیر (admin) دسترسی پیدا کنید. از آنجا می‌توانید روش‌های مختلفی برای اجرای کد از راه دور (RCE) انجام دهید، که این موضوع به عنوان تمرین برای خواننده باقی می‌ماند. این ضعف امنیتی نشان‌دهنده اهمیت بررسی و اصلاح اشکالات امنیتی در سیستم‌های احراز هویت و مدیریت نشست‌ها است.

نتیجه‌گیری

» CVE-2023-49105 به دلیل خطرات شدیدی که در پی دارد مانند دسترسی غیرمجاز به فایل‌ها، اجرای کد از راه دور از طریق ImageMagick و ارتقاء امتیاز به مدیر، بسیار اهمیت دارد.

» CVE-2023-49103 نشان می‌دهد که اطلاعات حساسی مانند تنظیمات PHPinfo قابل دسترس بوده و این موضوع می‌تواند به حملات احتمالی کمک کند.

هر دو CVE نشان می‌دهند که اصلاح و مدیریت موارد امنیتی در سیستم‌های احراز هویت و مدیریت نشست‌ها بسیار حیاتی است تا از سوءاستفاده و دسترسی غیرمجاز جلوگیری شود.

برای جلوگیری از آسیب‌پذیری‌های CVE-2023-49105 و CVE-2023-49103 که در اطلاعات ارائه شده مورد بحث قرار گرفتند، می‌توانید اقدامات زیر را انجام دهید:

برای جلوگیری از CVE-2023-49105

1. به‌روزرسانی ImageMagick

» اگر از ImageMagick برای تولید پیش‌نمایش تصاویر استفاده می‌کنید، مطمئن شوید که نسخه‌های استفاده شده از آن به‌روز و آخرین نسخه‌های امنیتی باشند. به‌روزرسانی دایمی این کتابخانه می‌تواند از حملات RCE که می‌تواند از طریق فایل‌های آپلود شده بوجود آید، جلوگیری کند.

2. تنظیم کلیدهای امضای URL برای همه کاربران:

» مطمئن شوید که همه کاربران دارای کلیدهای امضای URL منحصر به فرد هستند. این کلیدها باید به درستی پیکربندی شوند تا جلوی جعل یا استفاده نادرست از آنها توسط حمله‌کنندگان گرفته شود.

3. استفاده از مکانیزم‌های احراز هویت قوی‌تر:

» بررسی کنید که سیستم احراز هویت شما از مکانیزم‌های قوی‌تر و امن‌تری مانند احراز هویت دو مرحله‌ای استفاده می‌کند تا از هرگونه انتقال نادرست در اطلاعات احراز هویت جلوگیری کند.

برای جلوگیری از CVE-2023-49103

1. محدود کردن دسترسی به PHPinfo

» اطمینان حاصل کنید که PHPinfo فقط برای کسانی قابل دسترس است که نیاز دارند و برای عموم کاربران نه. محدود کردن دسترسی به این اطلاعات به کسانی که واقعاً نیاز دارند، می‌تواند به حفاظت از اطلاعات حساس PHP کمک کند.

2. اطلاعات حساس PHPinfo را پنهان کنید:

» ممکن است بخواهید PHPinfo را به جای افشای تمامی جزئیات، به شکل خلاصه و پوشش‌دهنده ارائه دهید. این کار می‌تواند به کاهش خطرات امنیتی کمک کند.

3. به‌روزرسانی و پیکربندی امن PHP

» همیشه از آخرین نسخه PHP استفاده کنید و اطمینان حاصل کنید که تنظیمات PHP شما به درستی پیکربندی شده‌اند و هیچ اطلاعات حساسی در دسترس عموم قرار ندارند.

نکات پایانی:

- مانیتورینگ و آپدیت روزانه: همواره مانیتورینگ راه‌اندازی کنید تا به زمان واقعی نگاهی به وضعیت امنیتی سیستم داشته باشید و آپدیت‌ها را به طور دوره‌ای اعمال کنید.
- آموزش و آگاهی کاربران: کاربران را آموزش دهید که به اشتراک گذاری اطلاعات احراز هویت و استفاده از ابزارهای امنیتی را به روز نگه دارند.

با انجام این اقدامات، می‌توانید از احتمال بروز CVEهای مذکور و دیگر آسیب‌پذیری‌های امنیتی در سیستم‌های خود جلوگیری کنید و امنیت آنها را تقویت کنید.

منابع

- https://ubuntu.com/
- https://www.ambionics.io
- https://nvd.nist.gov/


LIANGROUP.NET
021 9100 41 51 (300)