1.1.1 Gizlilik İhlali (Privacy Violation) (CWE-359) (CWE-200)

Açıklık Önem Derecesi: Orta

Açıklığın Etkisi: Kişisel kullanıcı bilgilerinin yetkisiz aktörlerce görüntülenebilmesi

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulamalar tasarımlarına göre kullanıcılarına ait kişisel veriler toplayabilirler, işleyebilirler, depolayabilirler veya farklı bir noktaya gönderebilirler. Bu kişisel verilerin topluca adına PII (Personel Identifiable Information) adı verilir. PII genel hatlarıyla bir bireyi doğrudan tanımlayan (örn; bir pasaport numarası, tc kimlik numarası, telefon numarası, eposta adresi, hesap bilgisi, parola, kredi kart numarası) veya dolayı yoldan ve daha yakından tanımlayan (örn; doğum tarihi, cinsiyet ve ikametgah) veri kümesidir. Daha genel manada tanımlanacak olursa PII sağlık verileri, korunaklı yasal kayıtlar ve dahasını içereren bir veri kümesidir. Bu verilerin açıktan sergilenmesi bireylere oldukça büyük zarar verebilir, çünkü bu veriler kimlik hırsızlığında, kişileri taklit etmede, gaspta ve kişiye fiziki zararlar vermede kullanılabilir. PII verileri çeşitli şekillerde işlenir ve tek bir global standardı yoktur. Fakat PII verileri iletim halinde korunmalıdır, mümkün olduğunca az açıktan sergilenir olmalıdır ve nadiren harici varlıklarla paylaşılmalıdır (Not: Hedeflenen sahip ile paylaşılabilir).

Uygulamalar PII verilerini uygulamanın dışına gönderdiklerinde (örn; bir yerel dosyaya, bir log'a veya harici bir web servise) veya PII verileri uygulamanın hata mesajlarında (örn; kullanıcıya dönülen hata mesajlarında, hypervisor veya docker gibi yönetimsel yazılımlar tarafından gözlemlenebilen terminal v.b. alanlara dönülen hata mesajlarında veya log'lara dönülen hata mesajlarında) yer aldığında sakıncalı olarak ele alınır. Bu duruma Gizlilik İhlali (CWE-359) (CWE-200) açıklığı adı verilir.

Log'ları ele alacak olursak uygulama PII verilerini log dosyalarına değiştirilmeden yazdırmamalıdır. Log'ların şifrelenmediği, periyodik olarak silinmediği ve erişimi uygun şekilde kısıtlanmadığı senaryolarda PII verileri yetkilendirilmemiş bir aktöre (örn; kötü niyetli bir geliştiriciye veya bir saldırgana) ifşa olabilir. Kötü niyetli bir geliştiriciye 2004 yılında bir AOL firması çalışanının 92 milyon müşteriye ait özel eposta adreslerini kumar web sitesine satması örnek olarak verilebilir.

Json Web Token'ları (JWT) ele alacak olursak uygulama PII verilerini Json Web Token'lara eklememelidir. Çünkü istemciye dönülen Json Web Token'lar istemci tarafta decode edilerek içeriği başkalarınca elde edilebilir ve PII verileri başkalarına ifşa olabilir.

URL'leri ele alacak olursak uygulama PII verilerini URL'lere eklememelidir. Çünkü URL bilgileri

- son kullanıcı bilgisayarındaki web tarayıcı önbelleğinde (cache'inde),
- son kullanıcı yerel ağındaki proxy sunucusunda (veya WAF'ta),
- hedef web uygulama sunucusundaki access log'larında (erişim log'larında) ve
- hedef web uygulama eğer dış (external) linklere sahipse Referrer http başlığı yoluyla diğer web uygulamaların sunucularındaki access log'larında (erişim log'larında)

yer alır. Bu ise bu konumlarda başkalarına PII verilerinin ifşa olması anlamına gelir.

Dosyaları ele alacak olursak PII verileri dosyalara olduğu gibi yazdırılmamalıdır. Çünkü PII verileri dosyaya erişen kötü niyetli bir geliştiriciye veya bir saldırgana ifşa olabilir.

Özetle bir kullanıcının kişisel verileri (PII) yetkilendirilmemiş bir aktör tarafından gözlemlenememelidir. Bu açıklığı somutlaştırmak için çeşitli teknolojilerde bazı örneklere yer verilmiştir.

Java - Güvensiz Kod Örneği:

```
// Kullanıcıya Bir Parolasının Geri Sızdırılması Gizlilik İhlali oluşturur.

public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();
    boolean isAuthenticated = session.getAttribute("isAuthenticated");
    if (isAuthenticated) {
        byte[] password = request.getParameter("password").getBytes();
        updatePassword(session, password);
        out.println("New password is " + (new String(password));
    } else {
        out.println("Authentication Failed");
    }
}
```

Güvesiz Java örneğinde kullanıcı parolası kullanıcıya geri gönderilmektedir. Bu sakıncalı bir kullanımdır.

C# - Güvensiz Kod Örneği (1):

```
// Kullanıcının parolası ekrana yazdırılmakta.

class PrivacyViolation
{
        static void CreateUser(string username, string password)
        {
                AddUser(username, password);
                System.Console.WriteLine(password);
        }
}
```

Güvensiz C# örneğinde kullanıcı parolası kullanıcıya geri gönderilmektedir. Bu sakıncalı bir kullanımdır.

C# - Güvensiz Kod Örneği (2):

```
// Kullanıcı Login Aktivitesini Parola ile Beraber Log'lama

User user = LoginUser(username, password);

if (user != null) {
    myLog.Debug(String.Format("Successful Login: {1}:{2}", username,
password);
}
else {
    myLog.Debug(String.Format("Failed Login: {1}:{2}", username, password);
}
```

Güvensiz C# örneğinde kullanıcı parolası log'lanmaktadır. Bu sakıncalı bir kullanımdır.

PHP - Güvensiz Kod Örneği:

```php
<?php

// Kullanıcının Epostası ve Parolasını Şifresiz Olarak Log'lama

if (isset($_POST['email']) && isset($_POST['password'])) {
    $email = $_POST['email'];
    $password = $_POST['password'];
    $loginSuccess = authenticate($email, $password);

    if ($loginSuccess) {
        // Sink
        error_log("$email : $password logged in successfully\n", 3, $logFile);
        header("Location: /account");
    }
    else {
        // Sink
        error_log("$email : $password incorrect credentials\n", 3, $logFile);
        header("Location: /login");
    }
    die();
}

?>
```

Güvensiz PHP örneğinde kullanıcı hesap bilgileri log dosyasına yazdırılmaktadır. Bu sakıncalı bir kullanımdır.

VbNET - Güvensiz Kod Örneği:

```
'Log Dosyasına Hesap Bilgilerini Şifrelemeden Yazdırma

Protected Sub WriteToLog(msg As String, user As String, pass As String)

    Dim strFile As String = "PATH_TO_LogFile\log.txt"
    Dim fileExists As Boolean = File.Exists(strFile)

    Using sw As New StreamWriter(File.Open(strFile, FileMode.OpenOrCreate))

        sw.WriteLine(DateTime.Now + ": " + msg)
        sw.WriteLine("User Details: " + user + ", " + pass)

    End Using

End Sub
```

Güvensiz VbNET örneğinde kullanıcı hesap bilgileri log dosyasına yazdırılmaktadır. Bu sakıncalı bir kullanımdır.

Javascript - Güvensiz Kod Örneği (1):

```
// Bir İstisnanın Fırladığı Hata Mesajında Hassas PII Kullanılması

app.post('profile/tcNo/update', (req, res) => {
    const { tc_no } = req.body;
    const user = getCurrentUser(req);
    if (validateSSN(tc_no)) {
        throw new Error(`${user.id} - Geçersiz TC No veya Zaten Bu TC No
Kullanımda - ${tc_no}`);
    }
}
```

Güvensiz Javascript kod örneğinde tc no PII verisi kullanıcıya hata mesajında geri döndürülmektedir. Bu sakıncalı bir kullanımdır.

Javascript - Güvenli Kod Örneği (1):

```
// Bir İstisnanın Fırladığı Hata Mesajında Herhangi Hassas PII Kullanılmaması

app.post('profile/tcNo/update', (req, res) => {
    const { tc_no } = req.body;
    const user = getCurrentUser(req);
    if (validateSSN(tc_no)) {
        throw new Error(`${user.id} - Geçersiz TC No veya Zaten Bu TC No`);
    }
}
```

Güvenli Javascript kod örneğinde tc no PII verisi kullanıcıya hata mesajında geri döndürülmemektedir. Bu güvenli kullanımdır.

Javascript - Güvensiz Kod Örneği (2):

```
// URL'de Hassas PII Gönderilmesi

function getUserInfo(user) {
    const first_name = user.first_name;
    const last_name = user.last_name;
    const tcNo = user.tcNo;
    const url = API_HOST +
`/api/get_info?fname=${first_name}&last_name=${last_name}&tcNo=${tcNo}`;
    fetch(url).then(
        // Handle response
    );
}
```

Güvensiz Javascript örneğinde tc no PII verisi URL'de GET parametresi olarak gönderilmektedir. PII verisinin istemciye geri gönderilmesi sakıncalıdır.

Javascript - Güvensiz Kod Örneği (3):

```
// POST Gövdesinde PII Gönderilmesi

function getUserInfo(user) {
    const data = JSON.stringify({
        first_name: user.first_name,
        last_name: user.last_name,
        tcNo: user.tcNo
    });
    const options = {
        hostname: API_HOST,
        path: '/api/get_info',
        method: 'POST'
    };
    const req = https.request(options, (response) => {
        // handlers for response
    });
    req.write(data);
    req.end();
}
```

Güvensiz Javascript örneğinde tc no PII verisi http paketlerinin gövdesinde (body'de) POST parametresi olarak gönderilmektedir. PII verisinin istemciye geri gönderilmesi sakıncalıdır.

Javascript - Güvensiz Kod Örneği (4):

```
// Kullanıcıya Dönülen Json Web Token'da PII'a Yer Verilmesi

const generateToken = (user, privatekey, options) => {
    const token = jwt.sign({
        name : user.name,
        role : user.role,
        address : user.address
    } , privatekey, options);
    return token;
};
```

Güvensiz Javascript örneğinde adres PII verisi Json Web Token başlığı içerisinde geri gönderilmektedir. PII verisinin istemciye geri gönderilmesi sakıncalıdır.

Javascript - Güvensiz Kod Örneği (5):

```
// Kullanıcı PII'larını Dosyalara Yazdırma

app.post('profile/new', (req, res) => {
    const { fname, lname, email, address } = req.body;
    // Kullanıcıları Dosyaya Log'lama
    fs.appendFile('users.csv',`\n"${fname}","${lname}","${email}","${address}"
`);
}
```

Güvensiz Javascript örneğinde eposta ve adres PII verileri bir dosyaya yazdırılmaktadır. PII verilerini dosyalara olduğu gibi yazdırmak sakıncalıdır.

Ruby - Güvensiz Kod Örneği:

```
pass = get_password()
...
dbms_logger.warn("#{id}:#{pass}:#{type}:#{tstamp}")
```

Yine aynı şekilde güvensiz Ruby örneğinde parola PII verisi log dosyasına çıktılanmaktadır. PII verilerini log dosyalarına çıktılamak sakıncalıdır.

Kurum uygulamada Gizlilik İhlali (CWE-359) (CWE-200) açıklığı tespit edilmiştir.

::::::::BULGU:::::::

Açıklığın Önlemi:

Bu açıklığın istismarını önlemek maksadıyla tavsiyeler şu şekildedir:

- PII verileri log'lara veya diğer dosyalara yazdırılmadan önce silinmelidir.
- PII verileri (veya herhangi bir türde hassas veri) kaydetmekten kaçınılmalıdır.
- PII verileri hata mesajlarının içerisine yazdırılmamalıdır.
- PII verileri iletim sırasında (in transit) ve depolama sırasında (at rest) şifrelenmelidir.
- Zorunluluk değilse PII verilerini log'larda, URL'lerde, hata mesajlarında, v.b. yerlerde açıktan sergilenir yapmaktan sakınılmalıdır.
- Zorunluluk varsa PII verileri yetkisiz durumlarda yalnızca kısmi gösterilecek şekilde ayarlanmalıdır. Örneğin PII verileri yalnızca ilk birkaç karakteri ile son birkaç karakteri

görülecek şekilde, fakat aynı zamanda hatanın da anlaşılmasına yardımcı olacak şekilde maskelenebilir.

- PII verilerinin uzak web servislerine gönderilmesinin gerekliliği ve gerekçesi tekrardan incelenmelidir ve sadece zorunluysa uygulanmalıdır.
- PII verilerinin haklı bir iş gereksinimi olmadan ve uygun şifreleme kullanılmadan uzak sunuculara gönderilmesinden kaçınılmalıdır.
- Json Web Token içerisinde PII verilerine yer verilmemelidir. Çünkü PII hassas verileri istemciye dönülen Json Web Token'a erişebilen herkese ifşa olmuş olacaktır.

Referanslar:

1. https://cwe.mitre.org/data/definitions/359.html
2. https://cwe.mitre.org/data/definitions/200.html
3. https://vulncat.fortify.com/en/detail?category=Privacy%20Violation
4. https://www.theregister.com/2005/02/07/aol_email_theft/