Web Uygulaması Güvenlik Denetimi ve Sızma Testi Raporu

Tarih: 16 Ocak 2026

Kapsam: Web Uygulaması Güvenlik Kontrolleri (SQLi, XSS, Mantıksal Hatalar)

Hazırlayan: Alperen Korkmaz

Durum: Tamamlandı

1. Yönetici Özeti

Bu rapor, hedef web uygulaması üzerinde gerçekleştirilen güvenlik testlerinin sonuçlarını içermektedir. Testler, düşük (Low) ve yüksek (High) güvenlik seviyeleri arasındaki farkları analiz etmek ve savunma mekanizmalarının etkinliğini ölçmek amacıyla yapılmıştır.

Temel Bulgular:

- **High (Yüksek)** seviyede uygulamanın SQL Injection ve XSS saldırılarına karşı **Prepared Statements** ve **Output Encoding** yöntemleriyle tam koruma sağladığı görülmüştür.
- **Mantıksal Zafiyet:** CAPTCHA modülünde, doğrulama sırasından kaynaklanan ciddi bir mantıksal tasarım hatası tespit edilmiştir.
- **Düşük Seviye Riskler:** AJAX/JSON endpoint'lerinde girdi denetimi eksikliği nedeniyle tam veri sızıntısı (Data Exfiltration) mümkündür.

---

2. Risk Derecelendirme Metodolojisi

Bulgular, OWASP standartlarına göre risk skoruna tabi tutulmuştur:

- **Kritik:** Doğrudan sistem ele geçirme veya tam veri sızıntısı.
- **Yüksek:** Hassas verilere erişim veya yetki yükseltme.
- **Orta:** Kısıtlı veri sızıntısı veya kullanıcı etkileşimi gerektiren saldırılar.
- **Düşük:** Bilgi ifşası veya düşük etkili zafiyetler.

---

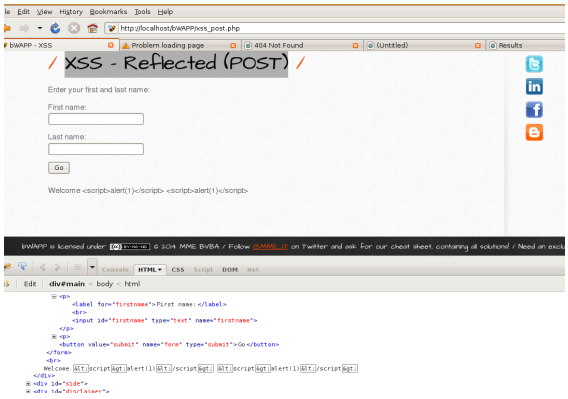
3. Ayrıntılı Teknik Bulgular

3.1 XSS – Reflected (POST)

Güvenlik Seviyesi: High

Durum: [PASSED] Güvenli

- **Saldırı Vektörü:** Kullanıcıdan alınan POST parametrelerinin (ad, soyad vb.) doğrudan HTML içine yansıtılması.
- **Teknik Analiz:** Yapılan denemelerde <script> ve <img> etiketlerinin tarayıcı tarafından yorumlanmadığı görülmüştür. Sunucu, tehlikeli karakterleri (reserved characters) HTML Entity formatına dönüştürmektedir.
- **Örnek Dönüşüm:**
    - Girdi: <script>
    - Çıktı: <script>



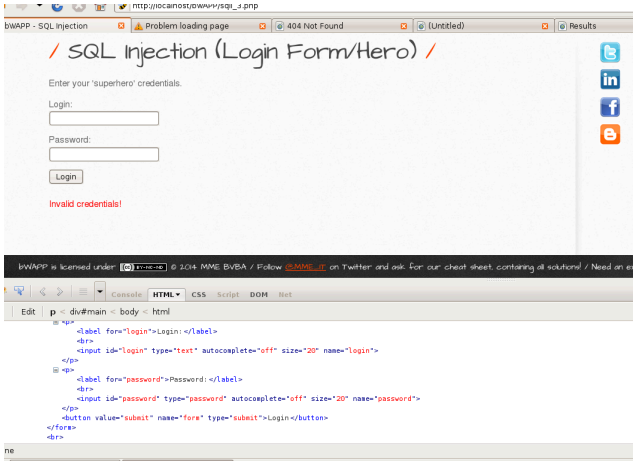
- **Kullanılan Koruma:** htmlspecialchars() veya benzeri bir "Context-Aware Encoding" kütüphanesi.

3.2 SQL Injection Analizi (Multiple Modules)

Güvenlik Seviyesi: High

Durum: [PASSED] Güvenli

Test edilen tüm standart modüllerde (GET/Select, Hero Login) SQL Injection denemeleri başarısız olmuştur.

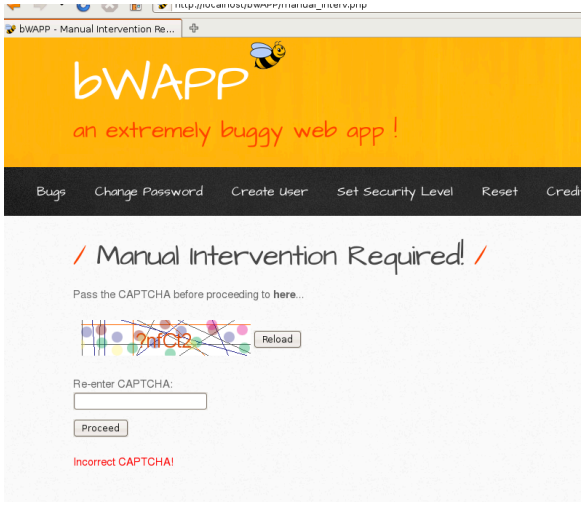
- **Teknik İnceleme:** Uygulamanın dinamik sorgu (String Concatenation) yerine **Parametrik Sorgu (Prepared Statements)** kullandığı teyit edilmiştir.
- **Mekanizma İşleyişi:**
  1. Sorgu taslağı veritabanına önceden gönderilir.
  2. Kullanıcı girdisi (' OR 1=1--) sadece bir "string" değer olarak işlenir, SQL komutu olarak değerlendirilmez.
  3. 

- **Sonuç:** Veritabanı katmanında tam koruma sağlanmıştır.

3.3 Mantıksal Zafiyet: Hatalı CAPTCHA Akışı

Güvenlik Seviyesi: High (Tasarım Hatası)

Risk Skoru: Yüksek

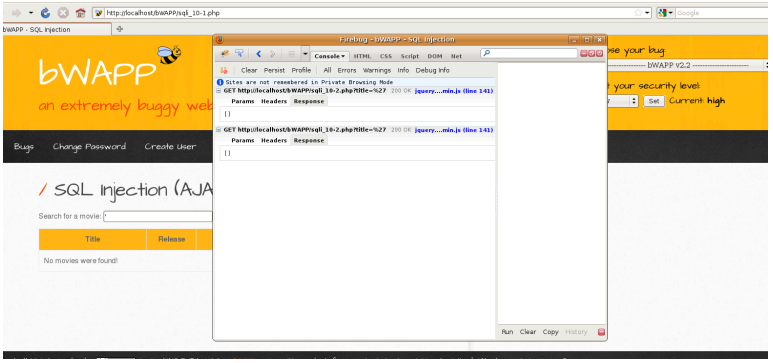
- **Açıklama:** CAPTCHA mekanizması, kaba kuvvet (Brute Force) veya SQLi saldırılarını engellemek için tasarlanmış olsa da, doğrulama sırası hatalıdır.
- **Hatalı İş Akışı:**
    1. Kullanıcıdan Girdi ve CAPTCHA alınır.
    2. **Hata:** Önce Veritabanı sorgusu (SQLi kontrolü/Giriş denemesi) çalıştırılır.
    3. Sonra CAPTCHA'nın doğru olup olmadığı kontrol edilir.
    4. 

- **Etki:** Saldırgan, CAPTCHA'yı yanlış girse bile arka planda SQL sorgusunu tetikleyebilir. Bu durum, CAPTCHA'yı tamamen işlevsiz bırakır.

3.4 AJAX/JSON SQL Injection

Güvenlik Seviyesi: high

Risk Skoru: Kritik

- **Açıklama:** Modern web mimarilerinde (AJAX/jQuery) arka plan isteklerinin (API endpoint) denetlendiği gözlemlenmiştir.
- **Bulgu:** yapılan işlemler sonucunda sql ınjection bypass edilmiştir sistem tarafından
- 

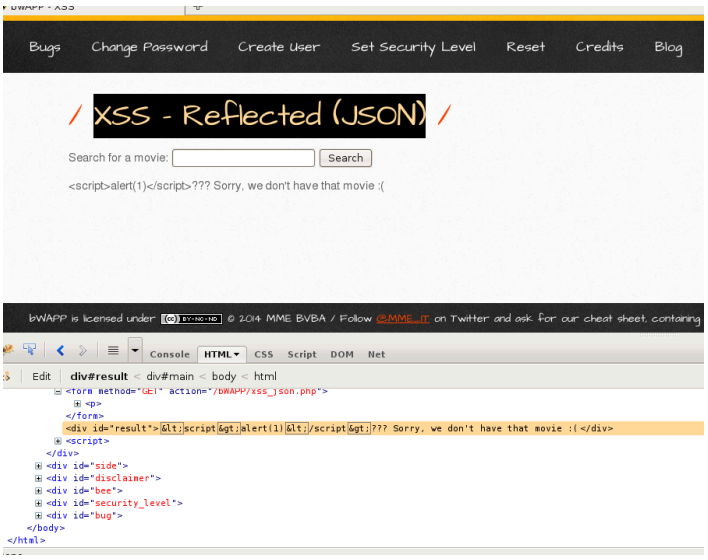
- **Etki:** bypass edilmesi sonucunda herhangi bir etkiye ulaşılmamıştır.

3.5 XSS - Reflected (JSON)

Güvenlik Seviyesi: High

Risk Skoru: Yüksek

Teknik Analiz: Yapılan denemelerde <script> ve <img> etiketlerinin tarayıcı tarafından yorumlanmadığı görülmüştür. Sunucu, tehlikeli karakterleri (reserved characters) HTML Entity formatına dönüştürmektedir.



Kullanılan Koruma: htmlspecialchars() veya benzeri bir "Context-Aware Encoding" kütüphanesi.

3.6 Xss - Reflected (get)

Güvenlik seviyesi: Low

Risk skoru: Kritik

Açıklama: Reflected XSS, kullanıcının tarayıcıya gönderdiği girdinin (query string parametresi gibi) sunucu tarafından işlenmeden veya güvenli hale getirilmeden HTML yanıtı içerisinde geri döndürülmesi durumunda ortaya çıkar.

Low seviye güvenlik yapılandırmasında:

- Kullanıcı girdileri sanitize edilmemektedir
- HTML karakterleri encode edilmemektedir
- JavaScript çalıştırılmasını engelleyen herhangi bir güvenlik kontrolü bulunmamaktadır

Bu nedenle saldırgan, URL içerisine zararlı JavaScript kodu ekleyerek uygulamanın bunu çalıştırmasını sağlayabilir. payload olarak <script>alert(1)<script/> kullanılmıştır. çalıştığının kanıtı aşağıdaki resimlerde mevcuttur.

3.7 Xss - Reflected (Back Button)

Güvenlik seviyesi: Low

Risk skoru: Kritik

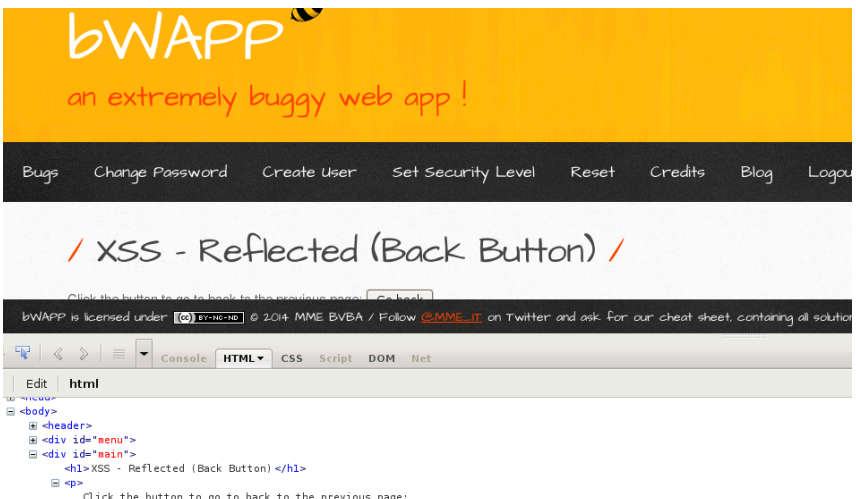
Açıklma: Back Button tabanlı Reflected XSS, tarayıcının önceki HTTP yanıtını önbellekten (cache) tekrar yüklemesi sırasında ortaya çıkar. Uygulama, önceki istekte kullanılan parametreleri yeniden işleyerek kullanıcıya gösterir ancak bu veriler herhangi bir output encoding işlemine tabi tutulmaz.

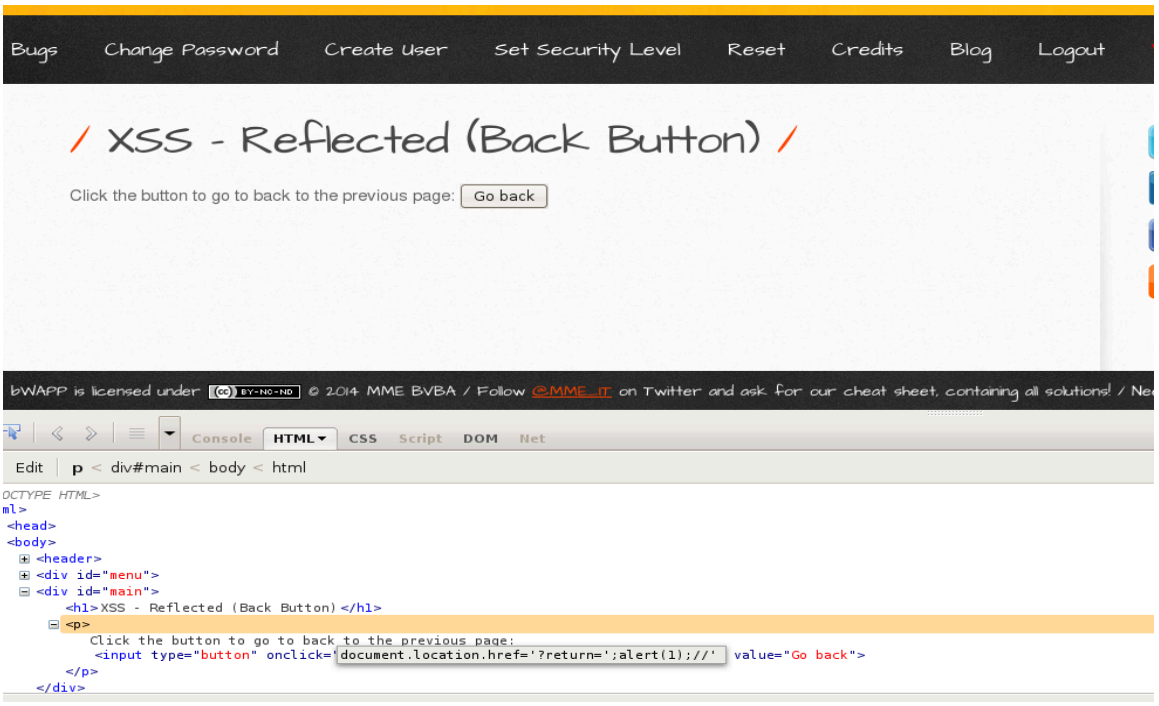
Low seviye güvenlik yapılandırmasında:

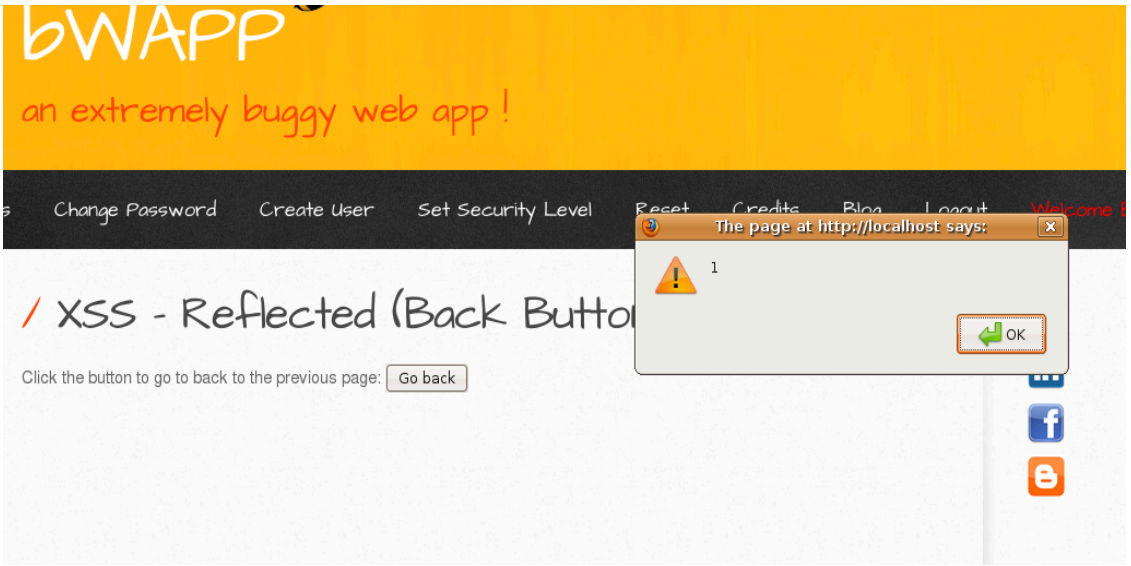
- Tarayıcı cache kontrolü yapılmamaktadır
- Kullanıcı girdileri filtrelenmemektedir
- HTML çıktısı güvenli hale getirilmemektedir

Bu nedenle zararlı JavaScript kodu, kullanıcı geri tuşuna bastığında tekrar çalıştırılabilir.

Bu nedenle saldırgan, URL içerisine zararlı JavaScript kodu ekleyerek uygulamanın bunu çalıştırmasını sağlayabilir. Payload olarak ?return=';alert(1);// kullanılmıştır. kod direk çalışmıştır. kanıt aşağıdaki görselde mevcuttur.




Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
Logout
/ XSS - Reflected (Back Button) /
Click the button to go to back to the previous page:
Go back
bWAPP is licensed under
© 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions!
Console
HTML
CSS
Script
DOM
Net
Edit
p < div#main < body < html
<h1>XSS - Reflected (Back Button)</h1>
Click the button to go to back to the previous page:
<input type="button" onclick="document.location.href='?return=';alert(1);//'" value="Go back">
</p>


bWAPP
an extremely buggy web app !
Change Password
Create User
Set Security Level
/ XSS - Reflected (Back Butto
Click the button to go to back to the previous page:
Go back
The page at http://localhost says:
1
OK

3.8 Sql Injection (Get / Search)

Güvenlik seviyesi: Low

Risk skoru: Kritik

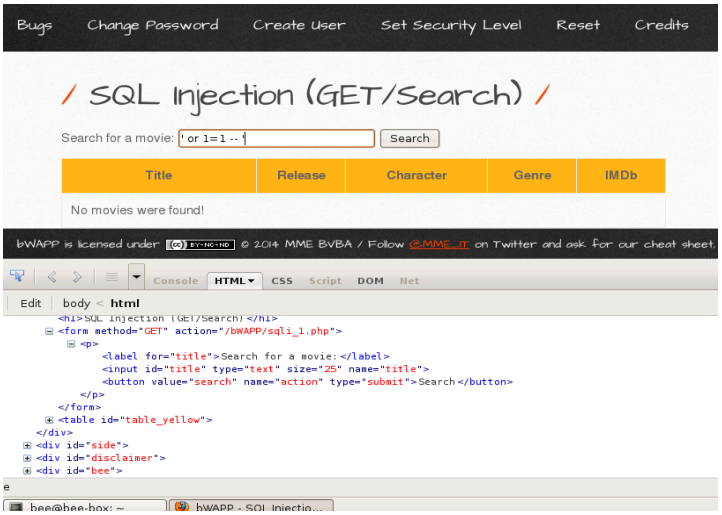
Açıklama: Arama (search) alanından gelen kullanıcı girdisinin:

Doğrudan SQL sorgusuna eklenmesi

Hiçbir filtreleme, kaçış (escaping) veya hazırlıklı sorgu (prepared statement) kullanılmaması nedeniyle ortaya çıkar. Bu da şu zafiyetleri doğurur.

- Tüm tablo içeriğini listeleme
- Veritabanı yapısını keşfetme
- Hassas verileri okuma
- UNION tabanlı enjeksiyonlara geçiş
- Bazı lablarda RCE'ye kadar zincirleme

Search alanına enjekte edilen zararlı kod direk çalışır bu lab da payload olarak (‘ or 1=1 - - ‘) kullanılmış ve bütün filmler önümüze dökülmüştür. kanıtları resimlerde mevcuttur.



3.9 Sql Injection (Post / Select)

Güvenlik seviyesi: Low

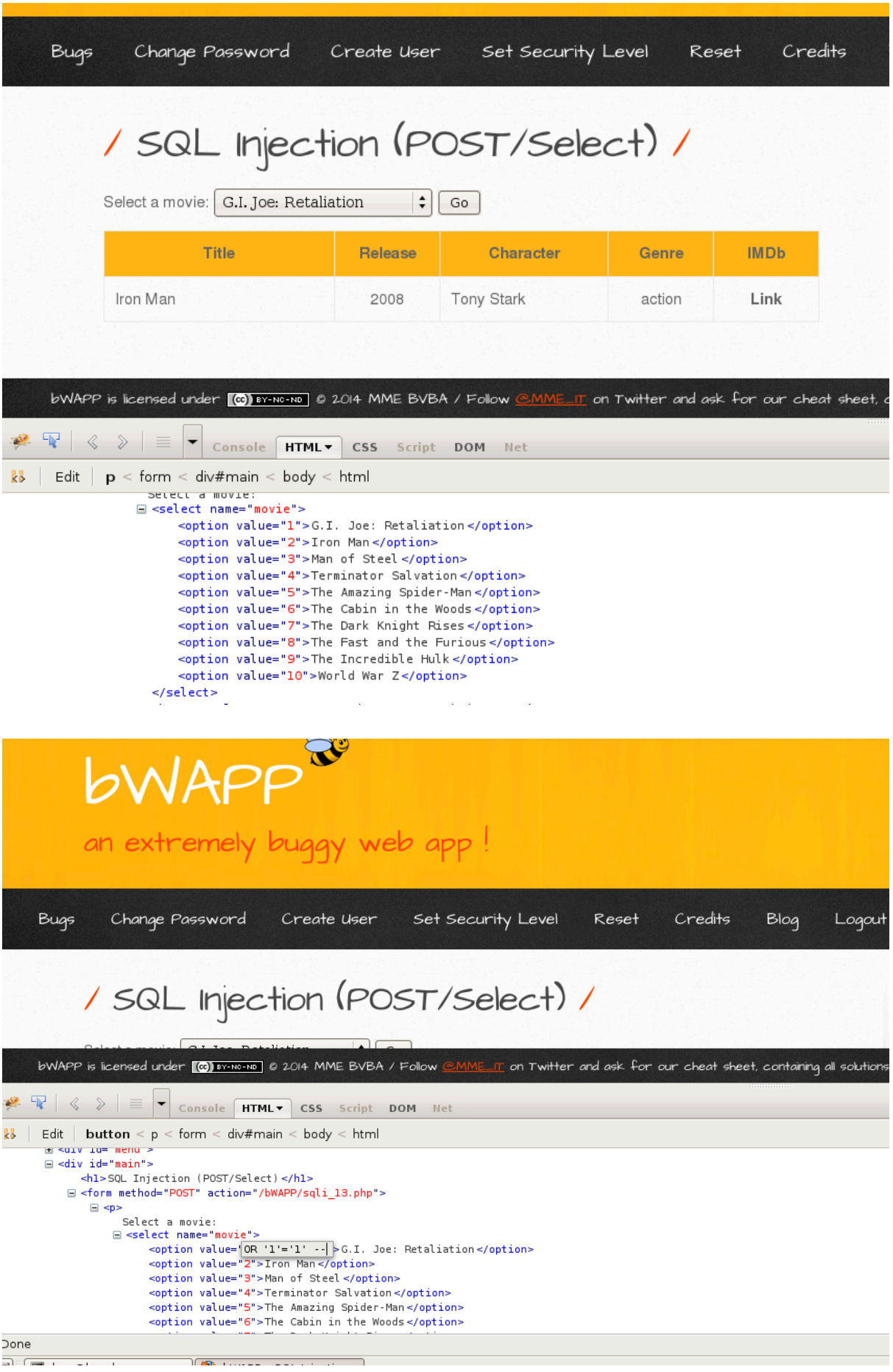
Risk skoru: Kritik

Açıklama: Kullanıcının bir form (dropdown, select box, radio button vb.) üzerinden gönderdiği verinin:

- POST isteğiyle sunucuya iletilmesi
- Bu verinin doğrudan SQL SELECT sorgusunda kullanılması
- Hiçbir filtreleme veya prepared statement uygulanmaması

sonucu ortaya çıkar. Kullandığım payload sadece kanıt için kullanılmıştır (‘ or 1=1 - - ‘)

bunun sonucunda error alınmıştır. Kanıtları resimler de mevcuttur.


Bugs
Change Password
Create User
Set Security Level
Reset
Credits
/ SQL Injection (POST/Select) /
Select a movie:
G.I. Joe: Retaliation
Go
Title
Release
Character
Genre
IMDb
Iron Man
2008
Tony Stark
action
Link
bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet,
Console
HTML
CSS
Script
DOM
Net
Edit
p < form < div#main < body < html
<select name="movie">
<option value="1">G.I. Joe: Retaliation</option>
<option value="2">Iron Man</option>
<option value="3">Man of Steel</option>
<option value="4">Terminator Salvation</option>
<option value="5">The Amazing Spider-Man</option>
<option value="6">The Cabin in the Woods</option>
<option value="7">The Dark Knight Rises</option>
<option value="8">The Fast and the Furious</option>
<option value="9">The Incredible Hulk</option>
<option value="10">World War Z</option>
</select>
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
Logout
/ SQL Injection (POST/Select) /
bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions
Console
HTML
CSS
Script
DOM
Net
Edit
button < p < form < div#main < body < html

<h1>SQL Injection (POST/Select)</h1>
<form method="POST" action="/bWAPP/sqli_13.php">
<p>
Select a movie:
<select name="movie">
<option value="OR '1'='1' --">G.I. Joe: Retaliation</option>
<option value="2">Iron Man</option>
<option value="3">Man of Steel</option>
<option value="4">Terminator Salvation</option>
<option value="5">The Amazing Spider-Man</option>
<option value="6">The Cabin in the Woods</option>
Done

---

4. Özet Bulgular Tablosu

Modül	Zafiyet Türü	Seviye	Durum	Risk
XSS Reflected	Cross-Site Scripting	High	Güvenli	Düşük
SQLi GET/Select	SQL Injection	High	Güvenli	Düşük
Login Form (Hero)	Auth Bypass	High	Güvenli	Düşük

Re-enter CAPTCHA	Mantıksal Zafiyet	High	**Zafiyet Var**	**Düşük**
AJAX/JSON Endpoint	SQL Injection	High	**Güvenli**	**Düşük**
XSS - Reflected (JSON)	Cross-Site Scripting	High	**Güvenli**	**Düşük**
Xss - Reflected (get)	Cross-Site Scripting	Low	**Zafiyet var**	**Yüksek**
Xss - Reflected (Back Button)	Cross-Site Scripting	Low	**Zafiyet var**	**Yüksek**
Sql Injection (Get / Search)	SQL Injection	Low	**Zafiyet var**	**Yüksek**
Sql Injection (Post / Select)	SQL Injection	Low	**Zafiyet var**	**Yüksek**

---

5. İyileştirme Önerileri (Remediation)

1. Mantıksal Akışın Düzeltilmesi (CAPTCHA)

Güvenlik kontrolleri "Fail-First" (Önce Hata Al) prensibine göre yapılmalıdır.

- **Öneri:** Kod bloğu, SQL sorgusuna gitmeden önce CAPTCHA'nın doğruluğunu kontrol etmelidir. Eğer CAPTCHA yanlışsa, veritabanı bağlantısı hiç kurulmamalıdır.

2. Prepared Statements Yaygınlaştırılması

Düşük seviyeli modüllerde görülen zafiyetleri gidermek için tüm SQL etkileşimleri parametrik hale getirilmelidir.

Örnek (PHP/PDO):
PHP
```
$stmt = $pdo->prepare('SELECT * FROM users WHERE id = :id');
$stmt->execute(['id' => $user_id]);
```

3. Savunma Derinliği (Defense in Depth)

- Sadece çıktı kodlama (output encoding) ile yetinilmemeli, girdi aşamasında da **Input Validation** (Beyaz Liste kontrolü) uygulanmalıdır.

- WAF (Web Application Firewall) kullanımı ile bilinen payload'lar ağ seviyesinde engellenmelidir.

---

6. Sonuç

Yapılan testler sonucunda, uygulamanın "High" seviye konfigürasyonunda temel enjeksiyon saldırılarına karşı dirençli olduğu, ancak “Low” seviye konfigürasyonunda temel enjeksiyon saldırılara karşı açık olduğu saptanmıştır.