

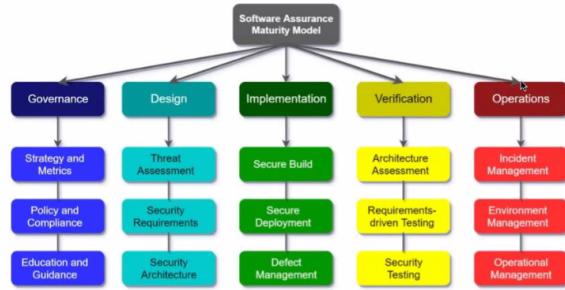
Kaynak Kod Analizi (Source Code Analysis)

OWASP SAMM (Software Assurance Maturity Model)

Geliştirilen uygulamaların, servislerin, betiklerin veya ürünlerin bilgi güvenliği bakış açısı dahilinde sağlam ve kaliteli olması isteniyor ise, yapılması gerekenler doğrultusunda planlanmalı ve geliştirme süreçlerinin her basamağına uygulanmalıdır.

SAMM, bilgi güvenliği felsefelerinin uygulama geliştirme süreçlerine entegre edilmesine yardımcı olmak için tasarlanmış bir model ve **yol haritasıdır**.

Uygulamalar, hangi kurumun hangi amaçla geliştirdiğine bağlı olarak farklı güvenlik sıkılık ve seviyelerine ihtiyaç duyarlar. Örneğin, açık kaynaklı olarak kullanılacak basit bir kütüphane uygulaması ile askeri bir iletişim yazılımının gereksinimleri, farklı olacaktır. Bu farklar SAMM Modeli üzerinde göz önüne alınmıştır.

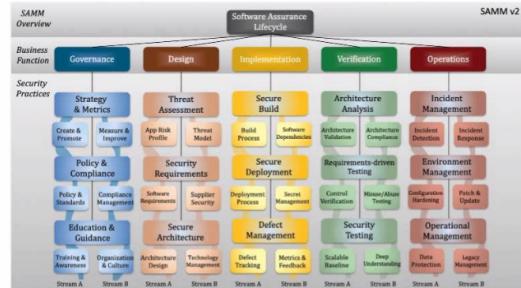


OWASP SAMM - Amaçlar

- Yazılım geliştirme süreçlerindeki güvenlik olgunluklarını bire bir takip etmek ve doğrulamak.
- Güvenli geliştirme için yapılacak incelemeleri belirlemek, alınacak aksiyonları planlamak, sorular sormak, gelişmeleri takip etmek ve denetlemek.

SAMM yapısı, **İş Modellerinden** ve **Güvenlik Pratiklerinden** oluşur. Üst başlıklar dahilinde 5 adet ana başlık bulunur. Bunlar; **Yönetim, Tasarım, Geliştirme, Doğrulama ve Yürütmeye**dir. Bu başlıklardan her birinin en az belli bir seviye kadarı, her kurum tarafından geliştirilen her türlü yazılım projesi dahilinde sağlanmalıdır.

Bu ana maddelerin altında bulunan **pratikler** ise, ilgili başlığın sağlanması için uygulanması gereken aktiviteleri belirtir. Her aktivitenin de kendi içinde sahip olduğu belli sağlananlık ve olgunluk seviyesi mevcuttur. Örneğin, Yönetim İş Modeli'nin sağlanması için Eğitim ve Destek Aktivitesinin tamamlanması, onun tamamlanması için ise altındaki **seviyelerin** tamamlanması gerekmektedir.



Yazılım Güvenliğine Giriş

OWASP SAMM - Amaçlar

Governance (Yönetim)

Strateji ve Metrikler: Bir plan olmadan güvenliği sağlamak zorudur ve çabalar, orantısız veya verimsiz olabilir. İş modelinin amacı, kuruluş dahilindeki yazılım güvenliği hedeflerini gerçekleştirmek için etkili bir plan oluşturmaktır.

Politika ve Uyumluluk: Yasal gereklilikleri temel olarak güvenlik ve uyumluluk kontrolleri oluşturmayı ve bu sayede yazılım güvenliğini artırmayı hedefler.

Eğitim ve Rehberlik: Yazılım geliştirme sürecindeki katılımcılar (proje yöneticileri, yazılım mimarları, geliştiriciler gibi) bilgi güvenliği bilinçlerini eğitimler ve teknik destek yoluyla kuvvetlendirmeyi hedefler.



OWASP SAMM - Amaçlar

Uygulama (Operations)

Olay Yönetimi: Yaşanan bir güvenlik olayını, ister kötü amaçlı ister ihmalkar davranış nedeniyle olsun, en az bir varlığın güvenlik hedeflerinin ihlali veya ihlal edilme tehdidi olarak tanımlar.

Çevre Yönetimi: Kuruluşun uygulama güvenliği konusundaki çalışmaları, uygulama hizmet vermeye başladiktan sonra sona ermez. Kullanılan teknolojiler eski hole gelene veya destekleri resmi olarak kaldırılana kadar, yeni güvenlik özellikleri ve yamalar düzenli olarak yayımlanır.

Operasyon Yönetimi: Sistem sağlama, yönetim, hizmet verme, hizmeti durdurma, veri tabanı temini ve yönetimi, veri yedekleme, geri yükleme, arşivleme ve benzeri operasyonlar, bu aktivite altında gerçekleştirilir.



Kaynak Kod Analizi

Araçlar - SAST

- SonarQube
- Coverity
- Snyk
- Appknox
- Veracode Static Analysis
- Fortify Static Code Analyser
- Codacy
- CodeScan
- HCL AppScan
- Checkmarx CxSAST
- Kiuwan Code Security & Insights
- Semgrep

SonarScanner-Cli

Windows VCG -> dil seç -> path seç -> scan de tarat

Fortify

MobSF

(OWASP sast tools list)

Security Code Scan - static code analyzer for .net - VisualStudio'ya bağlanabiliyor

Araçlar - DAST

- Netsparker
- Acunetix
- Appknox
- Veracode Dynamic Analysis
- Detectify Deep Scan
- Rapid7 InsightAppSec
- Checkmarx
- HCL AppScan
- GitLab Ultimate
- OWASP ZAP
- AppCheck
- Nikto
- Wapiti
- SkipFish

Karşıt Önlemler & Sıkilaştırma

Tavsiyeler

- **Encrypt It.** - Uzun vadede saklanacak ve aktif olarak kullanılmayacak veriler, şifrelenmelidir.
- **Hizmet Kapasite ve Kapasitesi** - Saldırı yüzey hacmi, uygulamanın kapasitesine bağlıdır.
- **Authenticated But Not Authorized** - Tanınmış, ama yetkilendirilmiş mi? / Her zaman kontrol.
- **Security Matters** - Güvenlik önemlidir.
- **Zero Trust Architecture** - Uygulama hiç kimseye güvenmemelidir.
- **Never Trust User Input / Validation** - Kullanıcı her zaman masum değildir.
- **Non-User Order** - API Sisteminin yapabileceği hiçbir işi kullanıcıdan emir alma.
- **Not-Yet Requests** - Bağımlılıkları ve iş sırasını her zaman kontrol et.
- **Integrity & SRI** - Dışarıdan geleni kontrol et.
- **Identifier Complexity** - Tanımlayıcıları karmaşıklaştır.
- **Endpoint Prediction** - Uç noktalar asla tahmin edilememelidir.
- **Parameter Obfuscation** - Verilen emirler anlaşılmamalıdır.
- **Generous Error Information** - Fazla bilgi zarar verir.
- **Limit The Rates** - Zaman/Metrik bazında düşün.

Uygulama Güvenliği & Hatalar

Geliştirici ve Kurum Hataları

- Deadline ve PoC aralıkları.
- Bütçe, envanter, insan ve iş gücü.
- Uygulamaların çalışması yeterlidir vizyonu.
- Hata vermedikçe hataya zorlanmaz.
- Exception Yönetimi genel tutulur.
- Yetersiz Q&A.
- Yetersiz Code Review.
- Stres Testlerinden kaçınırlar.
- Doküman değil, Blog'lar okunur.
- Github Issue & Security alanları atlanır.

Geliştirici ve Kurum Hataları

- Vulnerability Research yapılmaz.
- Copy & Paste kültürü.
- Eksik minimize, böl, parçala, yönet süreçleri.
- Sadece 'Authentication'.
- Full Trust Architecture - 'Biz bizeyiz' Kültürü.
- Trusting User Input, No Validation.
- Order From User.
- No Integrity & SRI.
- Basic Identifier.
- Basic Endpoint Naming.

WebApp ve Servis Yapıları

API ve REST API

API sistemleri, istemcilerden aldığı verileri kullanarak bazı işlemler gerçekleştiren, genellikle queu, veritabanı, dosya depoları ve SMTP servislerini aracılığına yönelik olarak harekete geçiren köprü mekanizmalarıdır.

- Embeded API
- REST
- SOAP

Monolitik bir bakış açısı ile incelendiğinde, API sistemleri barındıran uygulamaların güvenliği dahilinde öne çıkan ilk konu, **Client Side Filtering** olarak tanımlanabilir.

Teknik Zafiyet Yüzeyleri

API Security - OWASP API Security Project

API Nedir?

Application Programming Interface, çok basit ve genel anlatımı ile istemci uygulamaların, Web Sunucularından bilgi talep etmek veya onlara bilgi göndermek için kullandığı bir ekleni ve köprüdür.

OWASP Nedir?

Open Web Application Security Project, 1 Aralık 2001 tarihinde kurulan, kar amacı gütmeyen, güvenlik yazılımlarının hem bireyler hem de organizasyonlar için gelişmesine yardımcı olmak ve yazılım güvenliğini daha görülebilir kılmak amacıyla taşıyan bir organizasyondur.

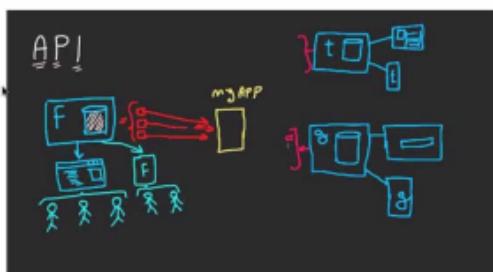
OWASP API Top 10 - API1:2019 Broken Object Level Authorization

Saldırıları, kendi istemci uygulamaların dahilinde API çağrısında bulunurken, kendi kullanıcı bilgilerini başka kullanıcılarla değiştirirler. Bu saldırının içinde, Web Sunucuların tarafında doğru yetkilendirme kontrollerinin olmaması durumunda, saldırıları belirtlen kaynağa erişimlerine izin verilmiş olunur.

Bu saldırının aynı zamanda Offensive Web Security tarafından; **IDOR (Insecure Direct Object Reference)** olarak da bilinir.

Saldırının engellenmesi için, aşağıdaki yöntemler entegre edilebilir;

- İsteklerin (API'lerin) sadece istemciler tarafından çalıştırılabilir olması.
- Tahmin edilemeyecek birleştirme 'Foreign Key'lerin kullanılması.
- API'ye gönderilen kimlik bilgilerinin düzensiz olarak yenilenmesi.
- Brute Force ve Dictionary Attack saldırısının önüne geçilmesi için WAF tarafından konfigürasyon yapılması.

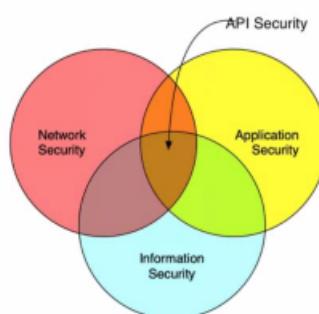


OWASP API Top 10 - API2:2019 Broken User Authentication

Yetkilendirme sistemlerinin sunucu değil, Client Side tarafı geliştirilmesi, İstemciler arası çatışlığı vakti yetkilendirme sistemlerinin de arası çatışlığı olmasına sebebiyet vermektedir. Bazı Frameworkler, API'ları ve anahtarları gizleyerek bunu engellemeye çalışma bile, zayıf yetek teknik olarak geçerliliklerini korumaktadır.

Örneğin JavaScript üzerinden yapılan bir kontrol sistemi, kullanıcı normalde yetki ihtiyacı olduğu bir sayfaya direkt olarak gitmeye çalıştığında kullanıcıyı giriş sayfasına yönlendiriyor ise, JS'yi dealfit eden bir saldırgan, yetkilendirmeyi **bypass ederek** dilediği sayfayı görüntüleyebilecektir.

- Cookie kullanılmayan uygulamalar dahilinde, kullanıcının horeketsiz darduğu belirli bir süreden sonra, sezonunun düşmemesi.
- Varsayılan veya dünya genelinde çok kullanılan parolaların kullanımına izin verilmesi.
- API sistemlerinin statik yetkilendirme anahtarları ile çoğaltılmış durumda bu anahtarların Front-end taraflarından elde edilebilmesi.



OWASP API Top 10 - API3:2019 Excessive Data Exposure

İstemiciler API'lara tanımlayıcı bilgilerini gönderdiklerinde, bu işlemi; belli bir veri setini veritabanından okuması için yapabilirler. Örneğin kullanıcı, kendisinin uygulamaya olan son 5 yanlış giriş tarihlerini teşep ettiğinde bu bilgilerin tamamı veri tabanından geliyor ve Client Side tarafından filtrelenerek ilk 5 odedi gösteriliyor ise bu, Web Sunucusunu gereksiz yere yoracaktır.

Bir başka örnek ile güvenlik tarafından bakılması gereklidir:

3D Secure işleminde Web Sunucusu tarafından cep telefonuna gönderilen bir SMS mesajı, API tarafından Client Side'a okunabileceği bir konuma geri gönderilir ve dolayısı ile kullanıcı tarafından girdilen girdinin bu gönderilen bu SMS değeri ile eşleşip eşleşmediği Browser Üzerinden kontrol edilir ise, saldırgan kişi bu elementi okuyarak 3D Secure sistemini atlatabilir.

Tüm kontrollerin Web Sunucusu tarafından halledilmesi, bu gibi problemlerin önüne geçecektir.



OWASP API Top 10 - API4:2019 Lack of Resources & Rate Limiting

Web Uygulamaları geliştirirken, kullanıcıların giriş bilgilerini ne kadar süre içerisinde kaç kere denediklerini sayan ve buna göre belirli engellemeye aksiyonları almaktan sonra kontrolcüler, genellikle **Client Side** olarak yerleştirilirler. Giriş bilgilerini API'ye göndermekle sorumlu olan input dinleyicileri üzerinde yapılan bu kontroller, her zaman yeterli olmamayırlar.

Intercept yöntemleri kullanarak API'ları direkt olarak yakalayan saldırganlar, eğer ki Web Sunucusu tarafından oluşturulmuş bir kontrol yok ise, **Brute Force** ve **Dictionary Attack** saldırısını gerçekleştirebilir ve bu şekilde sınırsız sayıda giriş denemesi gerçekleştirilebilirler.

Aynı şekilde saldırılar, gereklili kontrolün sunucu tarafında gerçekleştirilmemiş durumlarda, çok uzun girdi değerleri kullanarak API kaynak kodlarını bu değerleri veri tabanı üzerinde araştırması ile; sunucuya DoS saldırularına maruz bırakılabilirler.

Bu tür saldıruların önüne geçilmesi için hem API kaynak kodları, hem de WAF tarafından konfigürasyon yapılması gerekmektedir.



OWASP API Top 10 - API5:2019 Broken Function Level Authorization

Admin yetkisi ile veya test amoacı kullanılmakta olan API'lar; kullanıcılar sunulmakta olanlarla birlikte barındıldığı zaman, saldırganlar belli tahmin yöntemleri ile bunları bulabilmekte, böylece saldırı yüzeyini geliştirebilmektedirler. Tüm sistem tek API'ya bağlılığı zaman, bu sorunlar ile kolayca karşılaşılabilir.

IDOR zayıflığının ana farkı, parametrenin içindeki değişken değeri değiştirilmesi ile değil, direkt olarak çağırılan fonksiyonun değiştirilmesindedir. **Örneğin API**:

`get_client_data` yerine `get_staff_data`

veya

`/api/users/v1/myinfo` yerine `/api/admins/v1/ myinfo`

olarak çağırıldığında, yeterli kontrol sahibi olmayan bir uygulama, bilgi sizintisini sebebiyet verebilir.



OWASP API Top 10 - API6:2019 Mass Assignment

Öncelikle bilinmelidir ki Mass Assignment zayıflığı, çalışma sisteme göre; farklı yazılım dillerinde farklı isimlere sahip olabilirler.

Ruby on Rails, NodeJS: Mass Assignment
Spring MVC, ASP NET MVC: Autobinding
PHP: Object Injection

Bu zayıflık, basit örnekler ile anlatılabilir;

- Geliştiriciler, ihtiyaç duydukları zaman, performansı artırmak ve işlemleri hızlandırmak için sadece kaynak kodu dahilinde görülebilecek olan bazı Constructer tanımlarları. Bu değerler, kullanıcılar tarafından gelen verileri bulundukları sınıfların içerisindeki değişkenlere iletirler ve gerekli işlemleri gerçekleştirir.
- Aynı zamanda Framework ve yazılım eklientleri, geliştiricilerin işini kolaylaştırmak veya verileri sistemler arasında bozulmaması, farklı edilmeyenek şekilde taşımak için; bu verileri paketleyebilir veya objeleştirebilirler.



The screenshot shows a web browser window with the URL 'localhost:3001/users/new'. The page title is 'New user'. There is a single input field labeled 'Name' containing the value '1'. Below the input field is a 'Create User' button. At the bottom left of the page is a 'Back' link.

OWASP API Top 10 - API7:2019 Security Misconfiguration

Güvenlik eksik veya hatalı yapılandırmları, kendisine çok geniş bir kapsamı dahil etmeyece birek, bilinen belli başlı hataların yapılması veya **TODO'ların** dikkate alınmaması dolayısıyla; güvenlik zayıflarına davetiye çıkartılmaktadır.

- TLS** teknolojisi yerine eski ve zayıf barındırın **SSL** Sertifikalarının kullanılması.
- Bilgi sizintisi oluşturabilecek SQL hatalarının lettoğması.
- Hatalı bilgi girildiğinde, hangi bilginin hatalı olduğunu belirtmesi.
- API'ların Web Uygulamasının kaynak kodu içerisinde kolay bulunur şekilde barındırılması.
- HTTPOnly bayrak ve korumalarının kullanılmaması.
- API kullanımı için API Password yetkilendirmesi geliştirilmemesi.



OWASP API Top 10 - API8:2019 Injection

Eğer SQL, XSS, XXE ve yine Command Injection türü olan diğer parametreler, sadece Client Side tarafında kontrol ediliyor ise, saldırganlar belirli **Intercept** yöntemi ile ilgili API'ları yakaladıklarında, CI denemelerini direkt olarak API'lar üzerinden yapabılır ve böylece **CI** korumalarını **devre dışı** bırakabilirler.

Bu tarz saldırılardan korunmak için, görsellerde belirtilen ilgili yöntemlere başvurulabilir.

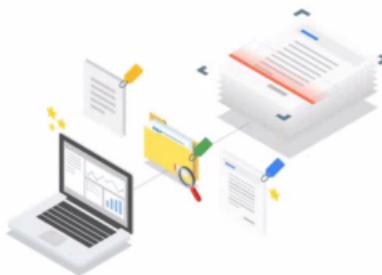
```
$data = $db->prepare( 'SELECT first_name,  
last_name FROM users WHERE user_id = (:id) LIMIT 1';  
);  
$data->bindParam( ':id', $id, PDO::PARAM_INT );  
  
PDO::PARAM_INT (integer)  
Represents the SQL INTEGER data type.
```

```
<?php  
if (isset($_GET['domain'])) {  
    echo "<pre>";  
    $domain = $_GET['domain'];  
    $lookup = system("nslookup $domain");  
    echo($lookup);  
    echo "</pre>";  
}  
?>  
  
Notice how the 'domain' parameter is taken in from the GET  
request, and immediately interpolated into a command string.
```

OWASP API Top 10 - API9:2019 Improper Assets Management

Dökümanlar yeni API versiyonlarını anlatacak şekilde güncellense veya eski dökümanlar sunumdan kaldırılsa bile, eski veya artık kullanılmayan API'lar devre dışı bırakıldığında, saldırganlar; internet ortamındaki uygulamalar arasında ve zamanında sunulmakta olan kaynak kodlarını veri tabanına yedekleyen bazı **3. parti uygulamalar** ile, **geçerliliğini yitirmiş** olan bu API'ları bulabilir ve sistemlere sızmak için kullanabilirler.

Zamanında sunucu tarafından sunulmuş ve **geçerliliğini yitirmiş** olan varlıkların yönetimi, bu açıdan önemlidir.



OWASP API Top 10 - API10:2019 Insufficient Logging & Monitoring

Yapılan işlemler, boşanlı olabileceği tahmin edilen saldırı bağlantıları ve sisteme zarar vermek amaç ile üretilen trafikler doğru şekilde takip edilmediğinde, yapılan saldırılardan engelleyici aksiyonlar almak, davranış analizi çıkartmak ve yapılan zararlı aksiyonlar sırasında bildirim olmak **mümkün olmayacaktır**.

Bu, saldırganların uzun süreler boyunca uzaklaştırılmışdan denemeler yapmasına izin verdiği gibi, saldırının yüzeylerinin de geniş olmasına sebebiyet verecektir.

Örneğin kısa sürede içerisinde yapılan **birçok giriş denemesi** hesap sahibi kullanıcuya bildirilmez veya karşı aksiyon alınmaz ise, saldırganın başarılı olma ihtimali günden güne yükselsecektir.

SOC ve **NOC** uygulamalarının gerçekleştirilemesi, bu gibi problemlerin önünü geçilebilmesi için gereklidir.



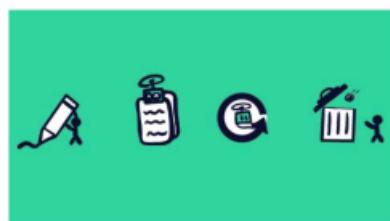
Ofansif Vizyonlar

Denetimsiz CRUD Yapıları

Bir çok uygulamanın ve API sisteminin Back-end操作ları, günün sonunda Create, Read, Update ve Delete mekanizmalarını çalıştmak için iş yaparlar. Bu操作lar veri tabanlarına bağlı olarak çalışabileceği gibi, sistem üzerindeki dosyalarla alakalı süreçler de yürütebilirler.

Günümüzde geliştiren bir çok uygulama, **Route** ve **Constructor** bazlı API sistemlerine sahip olmakla birlikte, **kullanıcılarından** elde edilen verileri **filtreleme** mekanizmaları olarak kullanmaktadır.

Tercih edilen bu yöntemler, varsayılan (out-of-box) haller dahilinde basit ve etkili manipülasyonlara açıklıklar ve Request'lerin işlenmesi sırasında doğru güvenlik önlemleri alınmaz ise, saldırganların zararlı aksiyonlarına izin vermektedirler.

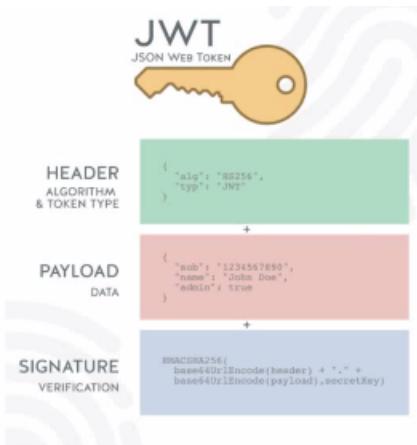


Token & Session Yapıları

QoS seviyesinin belli bir kalitede kalabilmesi için, uygulamalar, yetkilendirme kazanmış herhangi bir kullanıcıyı hatırla veya bir şekilde tanımak zorundadırlar. Günümüz web uygulamalarının mimarları geliştiriliyorken, sezon ve oturum yönetimi Stateless veya Stateful olarak tercih edilmektedir. Kendi aralarında farklı avantaj ve dez avantajlarına sahip bu yapıların güvenlik açıları da değişkenlik gösterebilir.

Load Balancing ve Mikroservis yapılarında daha rahat çalışabilen Stateless (Token Based (**SAML, SWT, JWT**)) yapılarında ortaya çıkan genel güvenlik problemleri, aşağıdaki gibi özetlenebilir;

- Okunabilir olması.
- Kaba Kuvvet soldırınlara yatkınlık.
- Token üretim uç noktalarının R/L teknolojileri ile korunmayışı.
- Expire sürelerinin gereğinden çok daha uzun olması.
- Token'ler içerisindeki veri ve parametre sızıntıları.
- Logout ve Parola değişimiyle etkilenmemeleri.

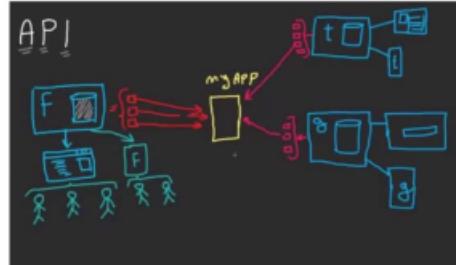


Not-Yet Requests

Uygulama dahilinde verilen hizmetler, farklı API uç noktalarına bağlı olarak çalışabilirler. Son kullanıcı tarafına gösterilen kontrol panelleri ve UI yapıları, yapılacak işlerin sırasını belirtirler ve kullanıcılarla adım adım ilerlemeleri için şablonlar sunarlar.

Eğer ilgili işlerin kendi arasında sırası ve birbirlerine bağımlılıkları var ise, aynı şablon mekanizmaları Back-end tarafından da belirlenmelidir.

Eğer saldırgan bir oktör tüm iş uç noktalarını keşfeder ve sistemleri soyut bağımlılıklara bağlı kalmadan rastgele olarak çalıştırır ise, henüz **hazır olmayan** mekanizmalar sisteme tüketim sağlayabilir veya bilgi sağlayan hatolar üretebilirler.



Bulk Information

API sistemlerine yapılan istekler, genellikle veri tabanlarını olmakta birlikte uygulamanın farklı fonksiyon ve parçalarından elde ettikleri bilgileri bir formatta düzenlemek ve Response içerisinde itetmek ile yükümlüdürler. Uygulamaların ve API'ların geliştirilmeleri esnasında **sadece ihtiyaç duyulan** doğru parametreler talep edilmez veya dönen cevaplar içerisinde **gereğinden fazla** bilgi depolanır ise, bu durum saldırganların hassas veya saldırı mimarlarının kurgulanmasında işe yarayacak verileri elde etmesine olanak sağlar.

Örneğin, bir kullanıcı hakkında bilgi talep edildiği zaman, sadece ihtiyaç duyulan verilerin toplanması ve itetilmesi gerekmektedir. Bunlar dışında kalınların da cevapta bulunması, saldırganların **uygulama mimarisini** haritalondurabilmesine olanak sağlar.



Hash Transferring

Veri tabanlarının ele geçirilme olasılıklarına ve dolayısı ile saldırganların verileri anlamlandırmamasına karşı güvende olması için, Görsel Blokaj, Hashing ve Salting operasyonları uygulanmaktadır. Bu durumlar, ele geçirilen verinin çözümlenmesini ve **Celart Text** olarak kullanılmasını engeller.

Teknik olarak incelendiğinde ise, bu çözümdeki bir veri tabanında yetkilendirme kontrolü yapmak için API tarafında kullanılan yetkilendirme anahtarlarının veri tabanındaki ile uyusması gereklidir.



Captcha Bypass

Captcha yapılan, son kullanıcıların gerçekliğini doğrulamak için kullanılır. Yapılanın implementasyonu CSRF Token sistemlerine benzer ve asıl amaç yapılan her bir veya birkaç istekte bir kullanıcının bilgisayar veya otomatik olmadığını doğrulamaktır.

Farklı varyasyonları bulunan ve uzun zamandır kullanılan Captcha sistemleri, farklı güvenlik problemlerine sahiptirler.

Sadece Front-end tarafında yapılan kontroller veya doğru entegre edilmemiş fonksiyonlar dolayısı ile, saldırgan tarafından çözülen bir Captcha'nın sağladığı yetkilendirme değeri, bir çok istek ile sürekli olarak kullanılabilir.

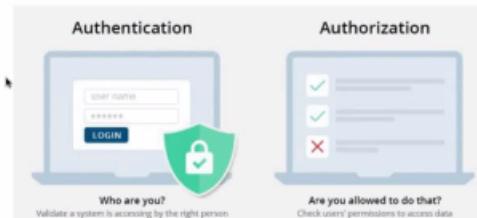
Sayı bazlı ve çözümü kolay olan görsel Captcha'lar, basit OCR yazılımları ile onlaşılmaktadır.

Matematiksel soruların bulunduğu Captchalar, basit Bot'lar tarafından algılanabilemektedir ve çözülebilmektedirler.



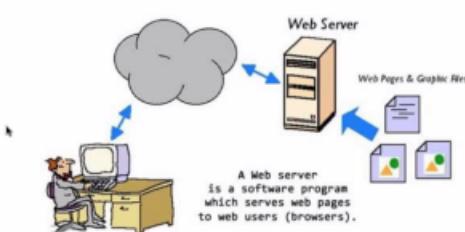
Yetkilendirme Problemleri

- Varsayılan hesaplar, tahmin edilebilir test kullanıcıları.
- Unutulan parolalar, erişim jetonları ve uç nokta bilgileri.
- Kontrol ve yönetim panellerinin erişilebilir olması.
- Test fazı sistemlerin kullanılabilmesi.
- Varsayılan doküman ve özelliğin aktif bırakılması ile yetkilendirme alanlarına yol gösterilmesi.
- Sistemlerin Brute Force saldırılara karşı korumasız olması.
- HTTP Redirection** durumlarının bilgi ifşaları sunması.
- Hassas bilgi sızıntıları sayesinde elde edilen değerler kullanılarak yetkilendirmelerin atlatılabilmesi.
- Uç noktaların herhangi bir yetkilendirme talep etmeden çalıştırılabilmesi.



Yük ve Request Yönetimi Hataları

- Redirection fazlarındaki bilgi ve kaynak kodu ifşaları.
- Kullanıcının haddinden ve gereğinden fazla karar veya emir verebilmesi.
- Istekler dahilindeki **bilgi ifşaları**.
- API uç noktalarının keşfedilmeyecek gibi kurgulanması.
- Uç noktaların hata yönetimlerindeki eksiklikler.
- Uç noktalar için çok basit isimlendirmeler kullanılması.
- Yük testlerinin gerçekleştirilmemesi.
- Rate Limiting** teknolojilerinin kullanılmaması.
- Reverse Proxy kullanımındaki eksiklik, isteklerin hörıcı API köprülerine tarayıcı tarafından İletimi.
- Tüm HTTP Metodlarının kabulü.



Oturum Problemleri

- HttpOnly bayraklarının kullanılması.
- Header yönetimi dahilinde yapılan hatalar.
- Secure bayraklarına ihtiyaç olmadığının düşünülmesi.
- Mobil uygulamalar ve kritik servisler dahilinde HPKP ve SSL Pinning yapılarının kullanılmaması.
- Session ve Token yapılarının hatalı yönetilmesi.
- ViewState değerlerinin şifrelenmemesi.
- Oturum zaman aşımlarının çok uzun tutulması.
- **HSTS** bayraklarının kullanılmaması veya kısa vadeli yapılandırılması.
- Tanımlayıcıların Auto Increment ve kısa tutulması.
- 'SameSite' değerinin kullanılmaması.
- Log-out durumunda yetkili jetonların sonlandırılmaması.



Continuous Pen-Testing

Sistemler, onlarında zararına kullanılan zayıflıklar ve bu zayıflıkları önlemek için üretilen teknolojiler devamlı olarak araştırılmakta ve geliştirilmektedir. Uygulama güvenliğinin düzenli, kararlı, **sürekli** ve verimli olarak takip edilebilmesi için, otomasyon ve manuel gerçekleştirilen sızma testleri ve zayıf analizleri uygulanmalıdır.

Bazı otomasyon zayıf analizi yazılımları, **Continuous** özelliklerini ve bulut tabanlı istihbaratları desteklemektedirler. Bu özelliklerin devreye alınması, uygulamaların güncel tehditlere karşı devamlı olarak korunmasına yardımcı olmaktadır. Otomasyon araçlarının tespit edemeyeceği bazı durum ve **mantıksal** yol haritaları ise, danışmanlar tarafından tespit edilmeli ve yarılma önerileri düzenlenmelidir.



Q&A, Bug Hunting, Bounty

Bir uygulamanın detaylı olarak sinanabilmesi için, zaman, ARGE ve spesifik incelemelerin gerektiği bilinen bir gerçektir.

Bu açıdan değerlendirildiğinde, otomasyon araçları veya danışmanlar tarafından gerçekleştirilen testlerin sahip olabileceği dezavantajlarından bazıları, büküş açılarının ve kullanılan tekniklerin **genel veya ezbere olabileceğidir**.

Uygulamaların henüz geliştirme aşamalarında iken kısa vadeli Q&A testlerine tabii tutulması, uzun vadede ise **Bug Hunting ve Bounty** kompanyalarına dahil edilmesi, farklı büküş açıları sayesinde güvenliğin daha verimli şekilde test edilebilmesine katkıda bulunacaktır.

