Enhancing Security in PHP
Best Practices and Tips

gokhan.cakirman@medianova.com

GÜVENLIĞIN ÖNEMI

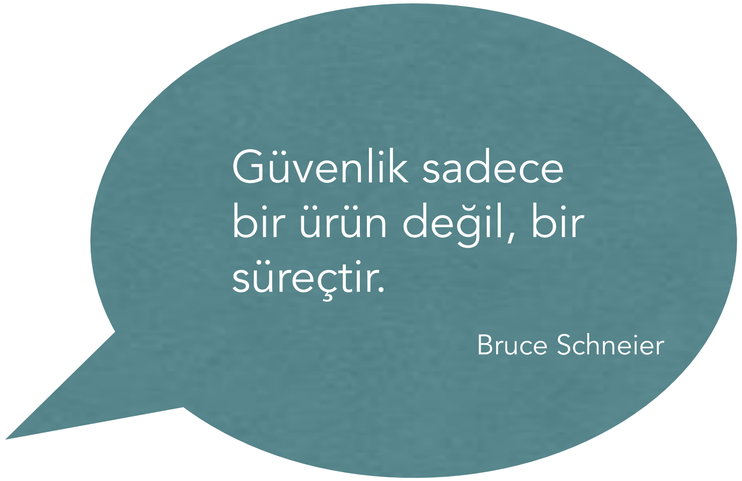
MEDIANOVA

- Güvenlik, hem bireylerin hem de kurumların dijital varlıklarını korumaları için hayati önem taşır.
- Kurumlar için müşteri bilgileri, çalışan kayıtları, finansal raporlar gibi hassas verilerin korunması zorunludur.

Verizon'un 2022 Veri İhlali Araştırma Raporuna göre, siber saldırıların %86'sı finansal sebeplerle gerçekleştirilmektedir.

MEDIANOVA

- Veri ihlaller, büyük miktarda mali kayıplara neden olabilir. (Cezalar, itibar kaybı, hukuki süreçler)
- Güvenlik zaafiyetleri aynı zamanda hizmet kesintilerine neden olabilir.
- Güvenlik, sürekli takip edilmesi ve güncellenmesi gereken bir süreçtir.



PHP VE GÜVENLIK

MEDIANOVA

- PHP, dünya çapında yaygın olarak kullanılan bir sunucu taraflı betik dildir.
- PHP’nin bu kadar yaygın kullanılması, doğal olarak çok sayıda hassas verinin bu dil ile işleniyor olmasını beraberinde getirir.
- Open source yapısı Php’nin geliştiriciler tarafından etkin bir şekilde kullanılmasına ve geliştirilmesine olanak tanır fakat aynı zamanda hackerlar için kolay bir hedef hale gelmesine sebep olur.

CVE (Common Vulnerabilities and Exposures)

PHP dahil olmak üzere çeşitli proglama dillerindeki bilinen güvenlik açıklarını listeleyen ve derecelendiren bir kaynaktır.

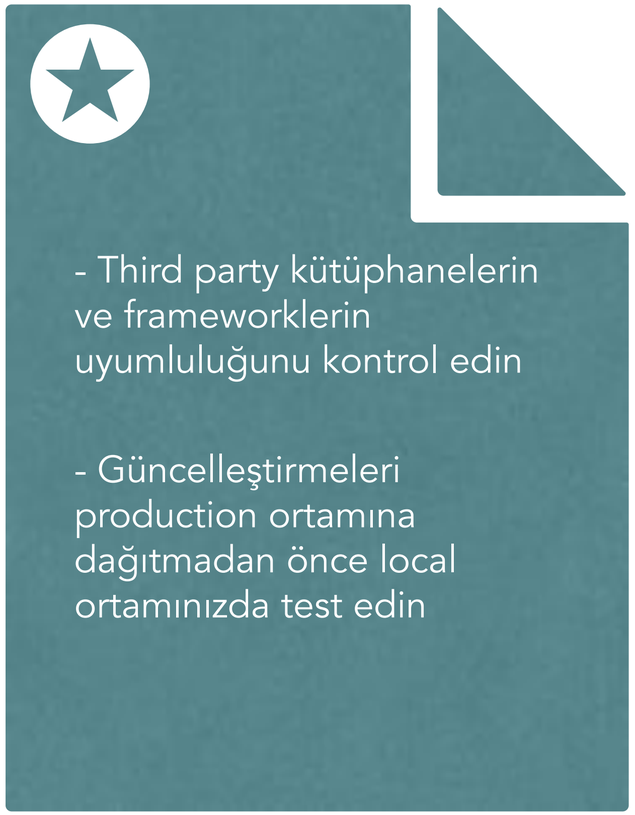
2024 PHP RAPORU

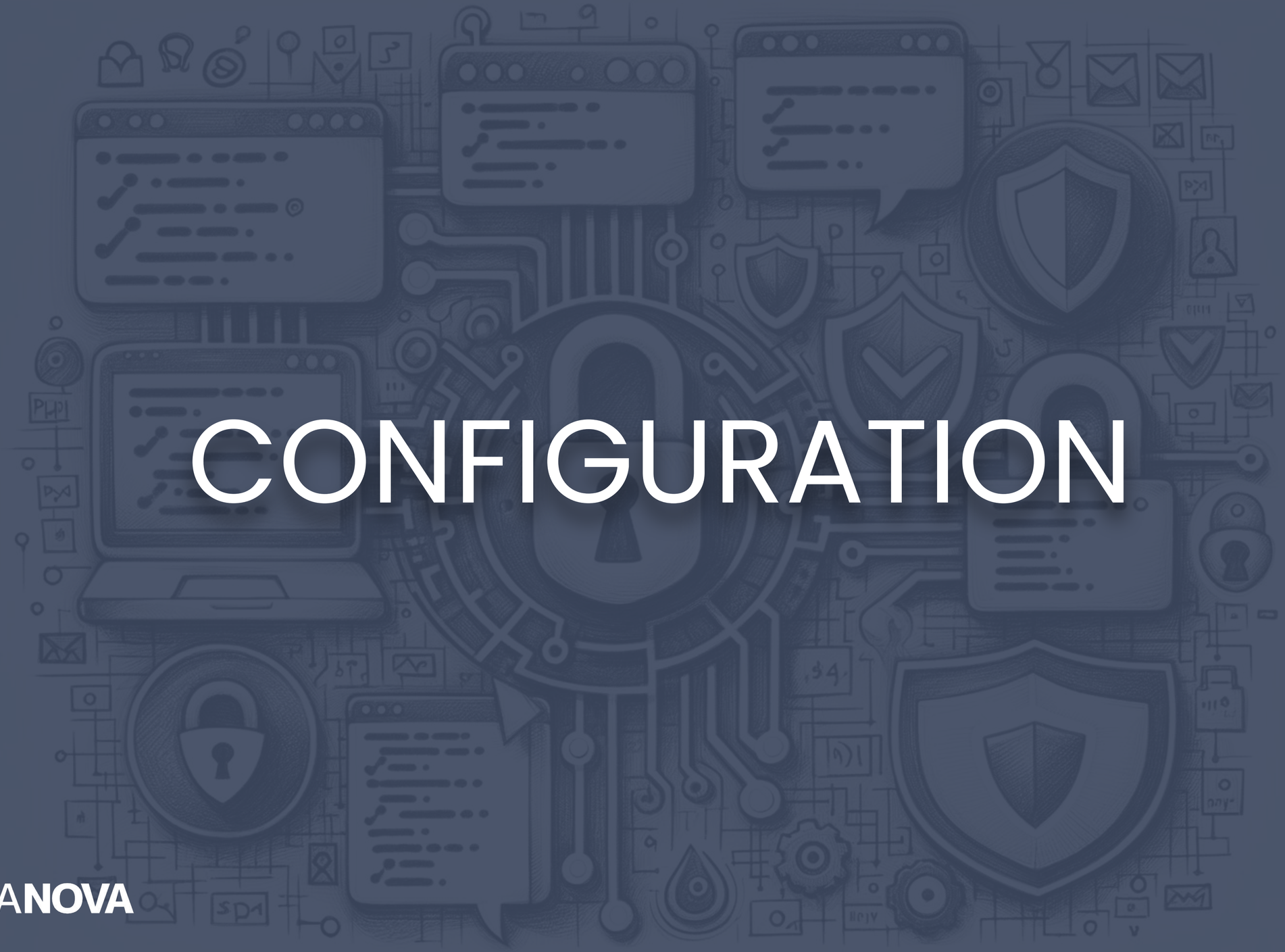
MEDIANOVA

GÜNCELLEŞTIRMELER

MEDIANOVA

- Her yeni sürüm, uygulamanızı potansiyel saldırılara karşı güçlendirir, bilinen sorunlara yönelik hata düzeltmeleri ve güvenlik düzenlemeleri içerir.
- PHP Version Manager(PVM), PHP FPM ve hosting kontrol panelleri gibi araçlar güncelleme sürecini kolaylaştırır.
- Yeni güncellemeler hakkında bilgi sahibi olmak uygulamalarınızı güncel tutmanızı ve yeni sürümün entegre edilmesi sürecindeki aksaklıkları önlemek açısından önem arz etmektedir.





CONFIGURATION

MEDIANOVA

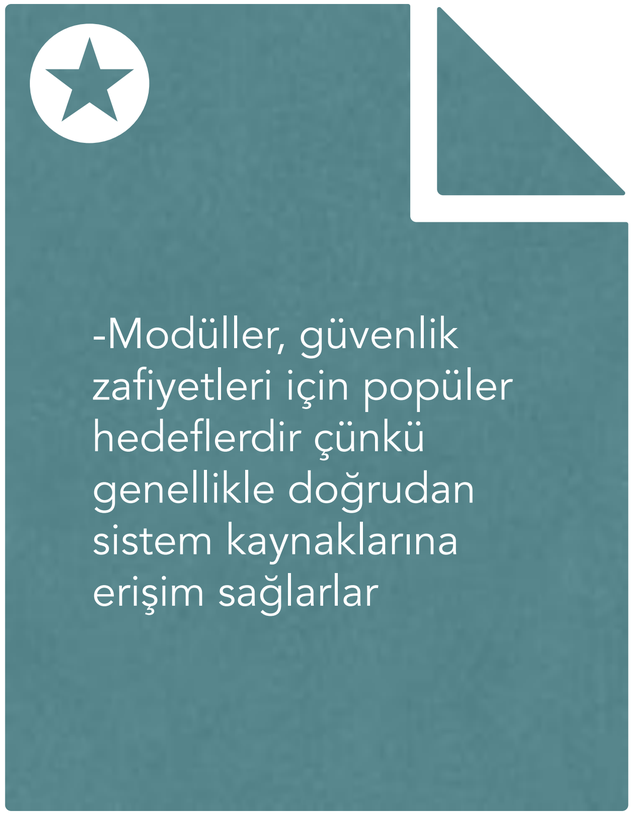
- Php konfigürasyonu, Php ortamınızın davranışını etkileyen çeşitli seçeneklerin ayarlanmasını içerir. Php kurulumlarının ana yapılandırma dosyası olan php.ini dosyası üzerinden kontrol edilir.

- Bu dosyadaki ayarlar, Php scriptlerinin nasıl çalıştıracağını, hataların nasıl ele alındığını, kaynak sınırlarını ve güvenlik gibi diğer yönleri belirler.

- PHP’nin doğru yapılandırılması, performansın optimize edilmesi, işlevselliğin sağlanması ve potanistel tehditlere karşı korunmak için hayati önem taşır.

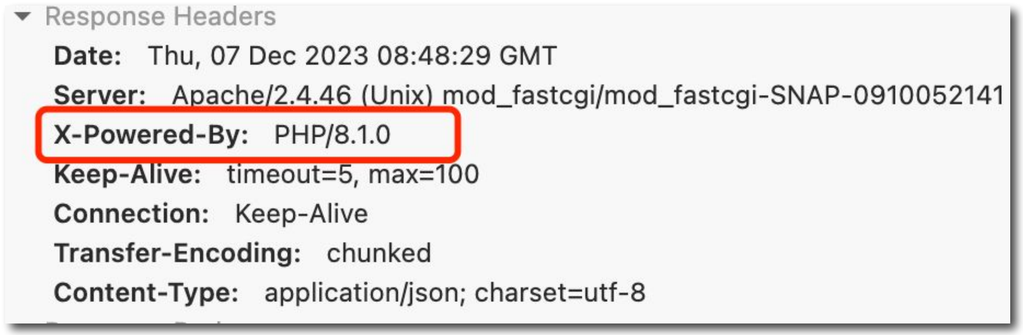
MODÜLLERIN KONTROL EDILMESI

- Her yüklenmiş modül, istismar edilebilecek potansiyel güvenlik açıkları ihtimalini arttırır.
- Kurulu PHP modüllerini listelemek için “php -m” komutunu çalıştırabilirsiniz.
- Hangi modüllerin projeniz için gerekli olduğunu,mevcut uygulamalarınızın ve kütüphanelerinizin gereksinimlerini inceleyerek belirleyebilirsiniz.
- “;extension=curl” php.ini güncellemesi ile modülleri kapatabilirsiniz.



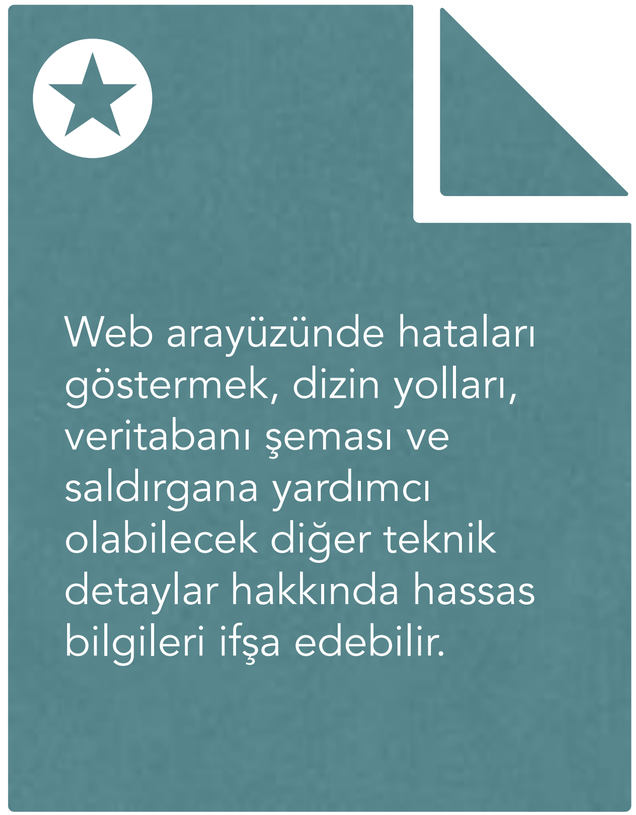
BILGI SIZINTISINI KISITLAYIN

- expose_php default olarak açıktır.
- Açık olması durumunda HTTP yanıtlarında header'da "X-Powered-By: PHP/version" ibaresi dönülür.
- Bu durum kötü niyetli kişilerin, o sürüme özgü bilinen zaafiyetleri hedef almak için gerekli olan bilgileri sağlamış olur.
- Php.ini içerisinde "expose_php = off" düzenlemesi yaparak bu özelliği kapatabilirsiniz.



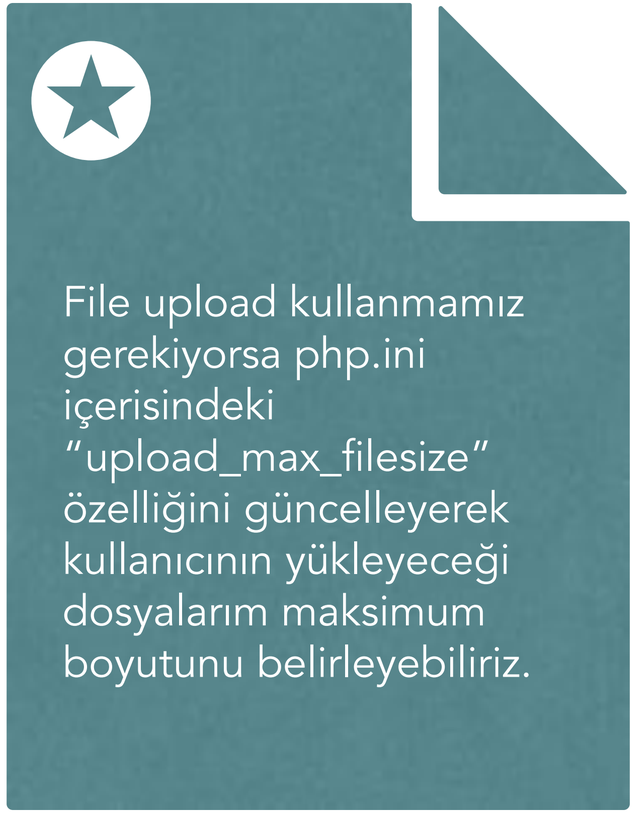
HATA GÖSTERIMINI KAPATIN

- PHP hatalarını kullanıcıya göstermek yerine tüm hataları Log dosyaları altına kaydetmek güvenlik açısından önemlidir.
- Hata gösterimini kapatmak için “display_errors = Off”
- Hataların loglanmasını sağlamak için “log_errors=On”
- Hataların belirlediğimiz path üzerindeki Log dosyalarına kaydedilmesi için “error_log = /path/to/your/error.log”



FILE UPLOAD ÖZELLİĞİNİN KAPATILMASI

- PHP'de dosya yüklemelerini devre dışı bırakmak, özellikle uygulamanızın bu işlevselliği gerektirmemesi durumunda önemli bir güvenlik önlemidir.
- Dosya yüklemesine izin vermek, sunucunuza kötü amaçlı dosyaların yüklenmesi ve çalıştırılması gibi risklere neden olabilir.
- File upload özelliğini kapatmak için php.ini içerisinde "file_uploads = Off" şeklinde düzenleme yapmamız yeterlidir.

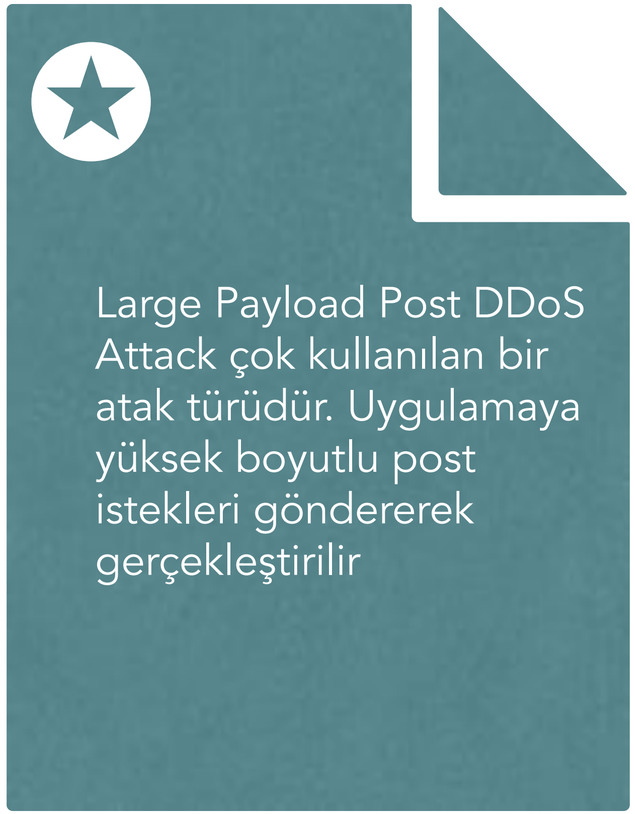


REMOTE CODE EXECUTION

- Remote code execution saldırıları, saldırganların sunucuda kendi kötü niyetli kodlarını çalıştırması ile gerçekleşir.
- Bu tip saldırılar genellikle gelen verilerin düzgün kontrol edilmediği ve uygulamamızın üss ataklarına açık olması durumunda tehlikeli olur.
- “allow_url_fopen” özelliği, Php’nin dosya fonksiyonlarının (file_get_contents, include, require) FTP veya web sitesi gibi uzak sunuculardan veri almasına izin verir.
- Php.ini üzerinde “allow_url_fopen=Off” ve “allow_url_include=Off” düzenlemelerini yapmamız sistemimizi çok daha güvenli hale getirecektir.

POST_MAX_SIZE

- Saldırganlar, sistem kaynaklarını tüketmek için aşırı büyük POST istekleri göndermeye çalışabilirler.
- Php.ini içerisindeki “post_max_size” özelliği ile maksimum post boyutunu düzenleyerek bu tarz saldırılardan kaçınmış oluruz.
- Bu ayar aynı zamanda dosya yüklemeyi de etkiler. Büyük dosyaları yüklemek için, bu değer upload_max_filesize değerinden büyük olmalıdır.



DIĞER KOMUTLAR

- “max_execution_time” her php scriptinin maksimum yürütme süresi
- “max_input_time” her komut dosyasının request datalarının parse işlemleri için haryabileceği maksimum süre
- “memory_limit” bir komut dosyasının tüketebileceği maksimum bellek miktarı
- disable_functions=exec,passthru,shell_exec gibi sistemimiz için tehlikeli olabilecek komutları kapatabiliriz.

GÜVENLI PROGRAMLAMA VE ATAKLAR

MEDIANOVA

VERI DOĞRULAMA VE SANITASYON

Veri doğrulama kullanıcılar ve diğer kaynaklardan gelen verilerin uygulama tarafından işlenmeden önce doğru, uygun ve güvelenir olduğundan emin olmayı amaçlar

Bazı doğrulama türleri:

- Tip Doğrulama
- Format Doğrulama
- Aralık doğrulama:
- Zorunlu alanlar

Sanitasyon: Veriyi zararlı içeriklerden arındırarak uygulama güvenliğini arttırma işlemidir.

XSS(CROSS-SITE SCRIPTING)

XSS saldırıları kötü niyetli kullanıcıların, hedeflenen web uygulamalarına zararlı scriptler eklemesine olanak tanır.

XSS Saldırılarını önleme yöntemleri

- htmlspecialchars() ,strip_tags(), fonksiyonlarını kullanın
- Web tarayıcısına, hangi kaynaklardan script yüklenebileceğini ve çalıştırılabileceğini belirten bir güvenlik katmanı eklemek için CSP kullanın.
- Çerezlere “HttpOnly” ve “Secure ”flag eklemek

SQL INJECTION

Sql injection, bir saldırganın veritabanı sorgularına kötü niyetli SQL ifadeleri ekleyerek veritabanını manipüle etmesini sağlayan bir güvenlik zaafiyetidir.

Sql injection önleme yöntemleri

- Prepared statements kullanın. Bu yöntem, girdi verilerin doprudan sql sorgusuna eklenmesini önler.
- Orm kullanarak veritabanı işlemlerini gerçekleştirmek, sql injection'a karşı bir koruma katmanı oluşturur.
- Her ne kadar hazırlanmış sorgular büyük bir koruma sağlasa da, giriş verilerinin sanitasyonu ve doğrulaması da önemlidir.

SESSION HIJACKING

Session Hijacking, saldırganın bir kullanıcının oturum çerezini (sessin cookie) çalarak o kullanıcının kimliğine bürünmesidir.

Session Hijacking için alınabilecek önlemler

- HTTPS Kullanımı
- Session içinde HttpOnly ve Secure bayraklarını kullanın.
- Oturum çerezlerine bir zaman aşımı süresi tanımlayın ve eski oturumları düzenli olarak sonlandırın.
- Session için ip kontrolü yapın ve farklı bir ip'den istek gelmesi durumunda oturumu sonlandırın.

SESSION FIXATION

Session Fixation, saldırganın bir oturum tanımlayıcısını (session ID) belirlemesi ve ardından kurbanın bu tanımlayıcı ile oturum saçmasını sağlamasıdır.

Session Fixation için alınabilecek önlemler

- Kullanıcı oturum açtığında veya yetkilendirme düzeyi değiştiğinde oturum ID'sini yenileyin. Bu işlem, session_regenerate_id() fonksiyonu ile yapılır.
- Oturum ID'lerinin tahmin edilmesi zor olacak şekilde rastgele ve yeterli yeterli uzunlukta olmasını sağlayın
- Session için ip kontrolü yapın ve farklı bir ip'den istek gelmesi durumunda oturumu sonlandırın.

CSRF

Cross-Site Request Forgery(CSRF), bir web uygulamasına yönelik ciddi bir güvenlik tehdidi olup, saldırganın kullanıcıların bilgisi dışında ve izni olmadan onların adına istek yapmasını sağlar.

Özellikle kullanıcılar sisteme oturum açmış durumda iken tehlikelidir

CSRF saldırıları genellikle kurbanın ziyaret ettiği bir web sayfası üzerinden kötü amaçlı bir link, resim, gizli form veya script aracılığı ile gerçekleşir.

Örneğin, kullanıcı bir bankacılık web sitesine oturum açmışsa ve başka bir sekmede CSRF içeren bir siteyi ziyaret ederse, bu kötü niyetli site kullanıcıları bankasına para transferi yapmasını sağlayabilir.

CSRF ATAKLARI İÇİN ALINABİLECEK ÖNLEMLER

- Token oluşturma ve doğrulama
- Çerezler için SameSite attribute eklenmesi
- Sunucumuza, gelen HTTP isteklerinin Referer veya Origin headerlarını kontrol ederek, isteklerin güvenli kaynaklardan geldiğinden emin olun .
- Özellikle hassas işlemler için, ek kullanıcı doğrulama adımları ekleyin.

PARAMETER TAMPERING

Bu saldırı türü, bir kullanıcının yada saldırganın uygulamanın işlevini etkilemek için Url, form verisi veya çerezlerdeki parametreleri değiştirmesiyle gerçekleşir.

Saldırgan bir web formundaki hidden, input veya select alanlarındaki verileri değiştirerek sunucuya gönderilen isteği manipüle edebilir.

Çerezlerde saklanan parametreler, kullanıcı tarafından veya kötü niyetli bir script tarafından değiştirilebilir. Bu değişiklikler, kullanıcının yetkilendirme düzeyini etkileyebilir ve uygulamanın davranışını değiştirebilir.

PARAMETER TAMPERING İÇİN ALINABİLECEK ÖNLEMLER

- Tüm girişleri, sunucu tarafından güvenilir bir şekilde doğrulayın ve sanitasyon işlemlerinden geçirin.
- Kritik parametrelerin kullanıcı tarafından değiştirilememesi için önlemler alın.
- ID yerine uuid kullanın
- Mümkünse URL parametlerini kullanmaktan kaçının. Verileri POST methodu kullanarak göndermeye çalışın.
- Güvenilir bir authorization yapısı geliştirin. Her işlemde kullanıcının bu işleme yetkisi olup olmadığını kontrol edin.

FILE UPLOAD

File upload işlemleri, web uygulamaları için değerli işlevsellik sağlasa da, yanlış yönetildiğinde ciddi güvenlik riskleri oluşturabilir.

Kötü niyetli dosyaları sisteme sızdırmak, sistem kaynaklarının kötüye kullanılması ve diğer güvenlik tehditleri gerekli önlemler alınmadığında gerçekleşebilir.

FILE UPLOAD ÖNLEMLERI

- Kullanıcı tarafından yüklenen dosya türünü (MIME tipi) kontrol edin sadece belirli dosya türlerine izin verin.
- Kullanıcıların yükleyeceği dosyaların boyutunu sınırlandırın.
- Yüklenen dosyaları doğrudan web sunucunun kök dizininden erişilebilecek yerlere kaydetmeyin ve erişimlerini kısıtlayın.
- Yüklenen dosya adlarını rasgele ve hash'lenmiş dosya adları ile değiştirin.
- Object storage gibi güncel teknolojileri kullanarak dosyaları kendi sunucunuzda barındırmamaya çalışın.

KIMLIK DOĞRULAMA VE OTURUM YÖNETIMI

MEDIANOVA

GÜÇLÜ PAROLA POLITIKALARI

Güçlü parola politikaları, hesapların yetkisiz erişimden korunmasında temel bir rol oynar. İyi tasarlanmış bir parola politikası, kullanıcıları güvenli parolalar oluşturmaya mecbur bırakır.

Güçlü Parola Politikalarının Özellikleri

- Parolalar için minimum karakter sınır belirleyin
- Parolaların büyük harf, küçük harf, rakam ve özel karakterler içermesini zorunlu kılın
- Kullanıcıların belirli periyotlarla parolalarını değiştirmesini isteyin.
- Belirli sayıda başarısız giriş denemesinden sonra kullanıcıyı geçici olarak kitlemek, brute force saldırılarını önlemeye yardımcı olur.

ÇOK FAKTÖRLÜ OTURUM DOĞRULAMA

Çok faktörlü kimlik doğrulama (MFA), kullanıcıların bir hizmete veya uygulamaya erişim sağlarken tek bir doğrulama yöntemine bağımlı kalmamasını sağlayan bir güvenlik sürecidir.

MFA'nın Üç Temel Faktörü

- Bilgi faktörü : Bu, genellikle bir parola, PIN kodu veya kişisel güvenlik sorularının cevapları gibi kullanıcının bildiği bir bilgi parçasıdır.
- Sahiplik Faktörü : Kullanıcının fiziksel olarak sahip olduğu bir obje üzerinden doğrulama yapılmasını içerir.
- Varlık Faktörü : Kullanıcının biyometrik özelliklerini içerir; parmak izi, yüz tanıma, ses tanıma veya retina taraması gibi kişiye özgü fiziksel veya davranışsal özellikler.

MEDIANOVA

API VE GÜVENLIK

MEDIANOVA

Güvenli API kullanımı, modern yazılım geliştirmede merkezi bir öneme sahiptir çünkü uygulamalar genellikle çeşitli hizmetler ve veri kaynaklarıyla etkileşimde bulunmak için API'ler üzerinden bağlanır.

API'ler, uygulamalar arasındaki veri alışverişini sağlayan arayüzlerdir ve bu nedenle güvenlik, API tasarımının ve kullanımının temel bir bileşenidir.

GÜVENLI API KULLANIMININ TEMEL BILEŞENLERI

1. Kimlik Doğrulama ve Yetkilendirme:
   - Kimlik Doğrulama: API'ye erişim sağlayan kullanıcıların veya servislerin kimliklerinin doğrulanması gereklidir.
   - Yetkilendirme: based access control veya fine-grained access control politikalarıyla sağlanabilir.
2. Veri Şifrelemesi:
   - Tüm API iletişimlerinin HTTPS üzerinden şifrelenmesi zorunludur. Bu, verilerin iletim sırasında dinlenmesini veya değiştirilmesini önler.
   - Hassas verilerin API yanıtlarında açık metin olarak gönderilmemesine özen gösterilmelidir.
3. Throttling:
   - Her kullanıcıya veya IP adresine saniyede veya dakikada izin verilen istek sayısı gibi limitler koyarak bu kontrol sağlanabilir.
4. Güvenlik Duvarı ve Erişim Kontrolleri:
   - API'leri kötü niyetli isteklerden korumak için API güvenlik duvarları ve diğer ağ güvenlik çözümleri kullanılmalıdır.
5. Hata Yönetimi ve Güvenli Loglama:
   - API'lerde hata yönetimi sırasında mümkün olduğunca az bilgi verilmelidir. Hata mesajları, iç sistem bilgilerini açığa çıkarmamalıdır.
6. API Dokümantasyon Güvenliği:
   - API'lerinizin nasıl kullanılacağına dair dokümantasyon yalnızca yetkili kullanıcılara açık olmalıdır.
7. Sürüm Yönetimi:
   - API'lerde yapılan değişikliklerin eski uygulamaları bozmadan yönetilmesi önemlidir.

GÜVENLI API KULLANIMININ TEMEL BILEŞENLERI

JWT, kimlik doğrulama ve bilgi alışverişi için kullanılan kompakt ve kendi kendine yeterli bir yoldur. Bir JWT, JSON nesnelerini şifreli bir yapıda saklayarak, bu bilgilerin güvenli bir şekilde bir taraftan diğerine taşınmasını sağlar.

Özellikleri ve Kullanımı:

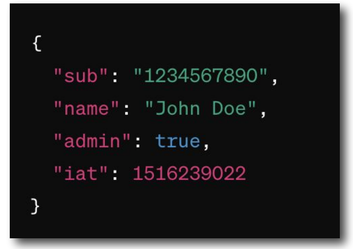
- Kompakt Yapı: JWT'ler, özellikle HTTP başlıkları üzerinden bilgi göndermek için kullanışlıdır çünkü küçük boyutludurlar.
- Kendi Kendine Yeterli: Bir JWT, gerekli tüm bilgileri içerir; bu, JWT'nin doğrulama bilgisi, kullanıcı bilgileri ve diğer metadata'yı barındırdığı anlamına gelir.
- Güvenlik: JWT'ler, genellikle HMAC algoritması kullanılarak veya RSA ile şifreli anahtarlar kullanılarak dijital olarak imzalanır.

JWT’NİN YAPISI

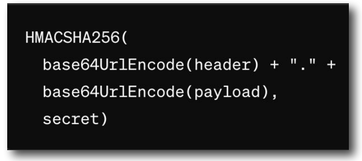
jWT header, payload ve signature olmak üzere üç ana bölümden oluşur

1. Header
   Header bölümü, token'in hangi türde olduğunu (JWT) ve hangi hashing algoritmasının kullanıldığını (genellikle HMAC SHA256 veya RSA) belirtir:

2. Payload
   Payload bölümü, token içinde taşınacak olan claim’lerini içerir. Bu claim'ler, token sahibi hakkında bilgiler veya diğer gerekli bilgiler olabilir:



3. Signature
   Signature, token'ın header ve payload kısmının, belirtilen algoritma kullanılarak bir secret key ile imzalanmış halidir. Bu imza, token'ın doğruluğunu ve bütünlüğünü doğrulamak için kullanılır.



MEDIANOVA

LOG STRATEJİLERİ

PHP API'lerinde güvenlik için etkili bir log yapısı kurmak, izleme ve güvenlik yönetimi için hayati öneme sahiptir.

Loglar, anormal davranışları tespit etme, sistem performansını izleme ve güvenlik ihlallerinin araştırılmasında kritik rol oynar.

İyi tasarlanmış bir loglama stratejisi, API'nizin güvenliğini artırmak için gerekli bilgileri sağlar.

LOG YAPISI NASIL KURULMALIDIR

1. Log Seviyelerini Belirleyin:
   Logları, bilgi (info), hata (error), uyarı (warning), ve hata ayıklama (debug) gibi çeşitli seviyelere ayırın.

2. Loglama İçin Araçlar Kullanın:
   PHP'de loglama için popüler araçlar arasında Monolog bulunur. Bu kütüphane, farklı loglama kanallarına (dosyalar, veritabanları, uzak sunucular vs.) destek sağlar ve esnek bir yapı sunar.

3. Zengin Context Bilgisi Ekleyin
   Log mesajlarına istek bilgileri, kullanıcı kimlik bilgileri, zaman damgaları ve hata mesajları gibi kontekst bilgileri ekleyin.

LOGLARI NERELERDE TUTABILIRIZ?

- Yerel Dosyalar: Sunucuda güvenli bir dizinde log dosyalarını tutmak basit ve erişimi kolay bir yöntemdir. Ancak, disk alanı ve güvenlik düşünüldüğünde sınırlamaları olabilir.

- Merkezi Log Sunucuları: Logları, ELK Stack (Elasticsearch, Logstash, Kibana) veya Graylog gibi merkezi log sunucularında saklamak, logları merkezi bir yerden yönetmeyi ve analiz etmeyi kolaylaştırır

- Bulut Tabanlı Loglama Hizmetleri: AWS CloudWatch, Google Stackdriver ve Azure Monitor gibi bulut tabanlı hizmetler, logları güvenli bir şekilde saklamak ve analiz etmek için geniş özellikler sunar.

LOGLARDA OLMASI GEREKENLER

- **Kullanıcı Kimlik Bilgileri:** Kullanıcı ID'si gibi kimlik bilgileri, hangi kullanıcının hangi işlemi gerçekleştirdiğini belirlemek için önemlidir.
- **IP Adresi ve Zaman Bilgisi:** Her log girdisinde isteği yapan kullanıcının IP adresi ve işlemin zaman damgası olmalıdır.
- **İstek ve Yanıt Bilgileri:** API isteklerine ve yanıtlarına ait bilgiler, hata ayıklama ve performans analizi için kritik öneme sahiptir.
- **Hata Mesajları:** Hataların ayrıntılı açıklamaları ve hata kodları, problemleri çözme sürecinde rehberlik eder.
- **Güvenlik İhlali İşaretleri:** Güvenlikle ilgili herhangi bir şüpheli aktivite veya ihlal belirtileri açıkça belirtilmelidir.

TOOLS

MEDIANOVA

- OWASP ZAP (Zed Attack Proxy): Açık kaynaklı bir güvenlik aracıdır ve özellikle web uygulamaları için penetrasyon testleri yapmak amacıyla tasarlanmıştır. Otomatik taramalar, pasif taramalar ve aktif taramalar yapabilir.
- Burp Suite: Burp Suite, profesyonel bir web güvenlik testi aracıdır ve güvenlik açıklarını tespit etmek için kapsamlı özellikler sunar.
- Acunetix: Acunetix, web uygulamaları ve API'ler için otomatik güvenlik taramaları gerçekleştiren bir araçtır. XSS, SQL Injection ve diğer birçok zafiyeti tespit edebilir.
- Nikto: Nikto, web sunucuları için açık kaynaklı bir güvenlik tarama aracıdır. Çeşitli dosya, CGI ve diğer güvenlik açıklarını tespit eder.
- SQLMap: SQLMap, SQL enjeksiyon zafiyetlerini otomatik olarak tespit etmek için kullanılan açık kaynaklı bir araçtır
- SonarQube: SonarQube, kod kalitesini ve güvenliğini izlemek için kullanılan bir platformdur. PHP dahil olmak üzere birçok dilde statik kod analizi yapar.

Sorular?

Teşekkürler.

MEDIANOVA

gokhan.cakirman@medianova.com