Web Uygulamalarında Güvenli Platform Seçimi

Ferruh Mavituna, Mayıs 2010, WGT E-Dergi 5. Sayı

Web uygulaması güvenliğindeki bir milyon klişeden biri de şudur:

Tüm sunucu-taraflı (server-side) diller aynı seviyede güvenlidir, her şey kodu yazan kişiye bağlıdır.

Eğer tüm yazıyı okumak istemiyorsanız hemen özetleyeyim, yukarıdaki laf yalandır, güvenlik ile kullanılan platformun (ASP.NET, PHP, Struts etc.) bire bir ilişkisi vardır.

İyi Programcı Güvenli Kod Yazar

Evet iyi bir programcı brainfuck ile 1 yılda güvenli bir kod yazabilir. Buna başlaması için ilk gereken şeylerden biri güvenli bir oturum (session) implemantasyonu yazmasıdır.

Eğer gidip bir web geliştiricisine hadi bakalım güvenli bir oturum sistemi geliştir de sonra da üyelik sistmini yazmaya başlayalım derseniz adam size güler, "*İşimiz gücümüz yok oturum' u biz mi yazacağız?*" diye ama aynı kişiye "Bizi CSRF'den koruyacak bir kod yaz da sonra diğer kısımlara başlayalım" derseniz paşa paşa yazmaya başlayacaktır. Platformların mevcut durumu göz önüne alındığında her programcının kendi CSRF korumasını yazması normal kabul edilir hale gelmiştir halbuki bu aynı oturum gibi platformun sorumluluğunda olması gereken şeylerden biridir.

Platformun Bizzat Kendisinin Güvenliği

Mükemmel bir platform yok, aynı bir çok uygulamada güvenlik açığı bulunduğu gibi bir çok platformda da güvenlik açıkları çıkıyor ama aynı bazı uygulamalar da daha az güvenlik açığı çıktığı gibi bazı platformda da daha az güvenlik açığı çıkabiliyor.

PHP bunun süper bir örneği. PHP'nin kendisinin Zend Hash Del Key Or Index Vulnerability gibi o kadar çok güvenlik açığı çıktı ki programcı ne kadar güvenli kod yazarsa yazsın sistemde hala güvenlik açığı olabilir.

Daha bir çok başka soru var. Platform NULL byte' ları olması gerektiği gibi işleyebiliyor mu? Unicode karakterlerde başı belaya giriyor mu? Cookie üzerinden gönderilen bir karakter yüzünde hata veriyor mu? vs. vs.

Platforma Özel Sorunlar

Bazı platformların bir kısmı varsayılan olarak güvenli tasarlanmıştır. Mesela ASP.NET' te HTTP Header Injection (CRLF/HTTP Response Splitting) pek görmezsiniz çünkü varsayılan olarak tüm ilişkili fonksiyonlar yeni satır kabul etmeyecektir. RFI' ı ASP uygulamalarında göremezsiniz çünkü basit şekilde böyle bir güvenlik açığa neden olacak bir yol yoktur.

Bunun gibi bir dizi platform özel güvenlik açığı ya da klasik yazılım geliştirme süreçleri vardır, bunlar da platformlara özel RFI gibi açıklara neden olurlar.

Aynı şekilde bazı platformlar salak özellikler ekler. Mesela Magic Quotes bunlardan biridir, nitekim en sonunda Magic Quotes'u PHP'den kaldırmaya karar verdiler.

Platformun Üzerinde Gelen Güvenlik Özellikleri

Sanırım herkes kendi şifreleme algoritmasını yazmanın ne kadar salakça bir iş olduğunu anlamış durumdadır ama buna rağmen herkes hala kendi XSS filtresini, CSRF korumasını, RegEx karakter escape' ini yazıyor ve bu bize çok doğal geliyor. Halbuki bu da aynı kendi şifreleme algoritmanızı yazmak kadar yanlış bir iş.

Bu yüzden platformunuzu değerlendirirken şu tip sorular sormanız lazım:

- Parameterized SQL Query'lerini destekliyor mu?
- HTML ve data arasındaki farkı algılayıp data'nın çıkış noktasına göre gerekli encoding'i yapıyor mu?
- Güvenli oturum desteği var mı?
- İşletim sistemi komutlarını parametre ve komutu kesin bir şekilde ayırarak güvenli bir şekilde çağırmaya izin veriyor mu?
- Email fonksiyonları birden fazla email adresinin eklenmesine, email header'larına yeni satır eklenmesine karşı bir güvenlik sağlıyor mu?

Maalesef hiç bir platform bunun gibi güvenlik için önemli işlerin hepsini yapamıyor ama bazıları kesinlikle gelinmesi gereken noktaya daha yakın.

Bu konuda daha fazla bilgi ve araştırma için Secure Web Application Framework Manifesto 'ı platform güvenliği konusunda güzel bir çalışma.

Dokümantasyon, Kültür vs.

Dokümantasyon, kültür ve etrafta dolaşan kod örnekleri de güvenlik noktasında ilginç bir rol oynuyor. Mesela Tomcat JSP ve IIS 6 ASP örneklerine bakabilirsiniz, tüm bu örnekler IIS ve Tomcat'in içerisinde geliyor ve içlerinde çok ciddi açıklar tespit edildi. Yani henüz daha işe yeni başlarken siz kod bile yazmadan güvenli olmayan bir sisteme sahipsiniz. Oradaki örnek kodları baz alarak yazılan kodlarda tabii ki benzer güvenlik sorunlarına sahip oluyor.

Güzel örneklerden biri de .NET dokümantasyonundaki bir çok kodda Parameterized SQL Query'leri kullanılıyor olmasıdır. Buna rağmen .NET, PHP ve diğer bir çok platform dokümantasyonda güvenli örnekler vermekten aciz bu da otomatik olarak yeni programcıların daha kötü kod yazmasına neden oluyor.

Güvenli Kod Yazmak İçin Gerekli Zaman, Efor ve Bilgi

Platformun tüm bu eksi ve artıları bir programcının uygulamasını güvenli hale getirmek için harcayacağı vakti çoğaltıyor, belli konularda daha deneyimli olmasını zorunlu kılıyor.

Eğer bir platform CSRF karşı uygulamayı güvenli getirme işlemini tek bir satır kod ile halledebiliyorsa o uygulama daha sade ve daha iyi analiz/test edilmiş bir kod ile piyasaya çıkacaktır. Bu da güvenlik kalitesinde anında bir yükselme demek.

Platform Önemlidir

Özetle platform seçimi bir web uygulamasının güvenliği konusunda kritiktir. Maalesef elimizde henüz çok iyi bir platform yok ama yavaş yavaş oraya doğru ilerliyoruz.

Makalede özellikle ASP.NET ve PHP örneklerine eğildim çünkü o iki platform ile daha yakından ilgiliyim. Bütün bu yazdıklarımı Ruby on Rails, Struts ve CppCMS gibi farklı platformlarda da gözlemleyebilirsiniz.