Web Uygulama Güvenliği Temel Kontrol Listesi

Bünyamin Demir, Mayıs 2010, WGT E-Dergi 5. Sayı

Web güvenliği ile direk veya dolaylı uğraşan herkesin belli bir aşamadan sonra sorduğu sorulardan biridir "Web güvenliği ile ilgili bir kontrol listesi var mıdır?". Aslında benim tecrübelerim web güvenliğinin daha fazla sezgisel olduğu, bu yüzden kontrol listeleriyle hareket edilen network güvenlik testlerine çok benzemediğidir. Tabi bazı durumlarda benzerlikler de olacaktır.

Web güvenliğinin sezgisel olmasınına dair en büyük kanıt; uygulamaların yazılış şeklinin veya yazıldığı ortamın çok fazla standardize edilmemiş olmasından kaynaklanmaktadır. Bir problemin çözümü için 100 yazılımcının, muhtemelen 5 farklı dil ile, 10 farklı yöntem sayesinde çözümü olacaktır. Zira network güvenliği için, bir çok şey protokoller üzerinden hareket ettiği için, insan faktörünün, zekasının veya insanın elinin içinde olmasının olayı çok farklı boyutlara götürmemesidir. Tabi bu cümlemi okuyan network güvenliği ile ilgilenen arkadaşların ateş püskürmemesini önemle rica ediyorum. Çünkü eğer root kullanıcısı ön tanımlı olarak ssh yapıyorsa, bunun kontrolü veya kapatması belli ve nerdeyse tek bir yoldan yapılabilir olması yanlış bir tanı olmayacaktır. Fakat web tarafında kimlik doğrulama gibi bir işlemin bir, iki, üç hatta daha fazla yoldan yapılabilir olabileceğini söylemek yanlış olmaz. Hatta aynı yöntemi kullanan ama farklı parametrelerle bunu daha fazla sıkılaştıran kişiler bile olacaktır. Bundan dolayıdır ki insan faktörünün güvenlik açısından en kritik olduğu yaşam bölgesi web olacaktır. Tabi web`in de üzerinde koştuğu network sistemini hiç bir zaman göz ardı etmemeliyiz. Sanırım bu sözlerim, network güvenliğinin biraz doymuşluğundan da kaynaklanıyor.

Sezgisel kısmı biraz daha açacak olursak; web karşımıza sadece banner, link ve janjanlı menülerden oluşan kısımların dışında, arka planda çok fazla girdi, çıktı ve hatta bu girdi çıktı mekanizmasını sağlayan birden fazla fonksiyonun birleşmesi sonucu oluşan kompleks yapının kendisi olmasıdır. Bu yüzden bir kişinin aynı uygulama için farklı zamanlarda bulacağı güvenlik açıklıkları sayıca ve kıritiklik bakımından bir birinden farklılıklar gösterebilmesi söz konusudur. Network güvenliğinde ise yapı değişmediği sürece ve güvenilir bir kontrol listeniz varsa, kontrol ettiğiniz maddeler değişmediği sürece bulunan zaafiyetlerin sayısı ve kritiklik seviyeleri çok fazla değişmeyecektir. Sanırım web`in network yapılarına göre daha fazla canlı ve değişken olması da bu tezi destekleyecektir.

Yine de tüm bu "kontrol listesi yoktur" düşüncelerime rağmen, kabaca bir web güvenlik testi kontrol listesi yazacak olursak, muhtemelen aşağıda belirteceğim başlıklar olacaktır.

1. SQL injection var mı?
2. XSS var mı?
3. İş mantığı problemi var mı?

Tüm kontrol listesinin %80`ini bu üç madde oluşturur desem güler misiniz? Hatta buyrun bu testler için neler yapılabileceğine bakıp, daha fazla gülelim.

1. SQL injection denetimi?

Bir çok kişinin ilk aklına gelecek olan ‘ (tek tınak) parametresini girdinin sonuna ekleyip göndermesi ve eğer hata alıyorsa; “evet, burda SQL injection olabilir” demesidir. Gördüğünüz gibi “olabilir” dedik. Dolayısıyla bu üç madde için ilk testlerimiz çoğunlukla 1-0 arasında bir değer alacaktır. Ne kesin olarak “Vardır” diyebiliriz, ne de “Yoktur”. Bir diğer sıkıntı ise, evet bir parametre için kontrol ettik ama ya diğer tüm parametreler için? Sanırım orta ölçekli bir web uygulamasının binlerce girdi noktası olacaktır.

2. XSS var mı?

XSS denetimi için girdi yerine “<script> alert(1);</script>” yazdık ve karşılığında herhangi bir karmaşıklık görmedik. Dolayısıyla “XSS vardır” demek güçtür. Fakat bu test ile “XSS yoktur” demekte bir okadar güçtür. Dolayısıyla 1-0 arasındaki değer ile birlikte test parametresinin doğruluğu hatta birden fazla test parametresinin gerekliliği söz konusu olacaktır. Yine SQL injection da olduğu gibi, sadece bir parametre için değil, tüm bulunan parametreler için geçerli bir test yapmak gerekecektir.

3. İş mantığı problemi var mı?

Çok genel bir terim olduğunu söyleyebilirim. Hatta bilinen açıklıkların %80`i bu açıklık altında da kategorize edilebilir. Zira iş mantığı problemi, kişinin hatalı kodlamasından kaynaklanan problemlerdir –ki XSS ve SQL injection da bu tür hatalı kodlamalardan kaynaklanmaktadır. Testi diğerlerine göre çok daha zordur. Çünkü test için sabit bir parametresi yoktur. Bunun yerine test yapılacak form/girdi v.s alanlarının hangi iş modeli için tasarlandığını bilmek ve bunun aksi yönde hareketlerde bulunup, ortaya çıkabilecek açıklıkları raporlamaktır. Dolayısıyla bunun için bir kontrol listesi oluşturmak zor.

Tüm bu söylemlerime rağmen yine de özellikle kullanılan metodun kontrolünü amaçlayan bir kontrol listesi hazırlayalım. Tabi bu listeye farklı dillerin kendine özgü zaayifetleri de eklenebilir. Fakat ilk aşama da hepsini listeleyemesekte, kabaca aşağıdakine benzer bir listemiz olacaktır.

1. Login esnasında username ve password parametreleri HTTPS üzerinden gitmelidir.
2. COOKIE için de password, kullanıcı adı gibi bilgiler olmamalıdır.
3. Kullanılan şifre algoritması yeterince katı mı? (En az bir büyük harf, en az bir küçük harf, en az bir sayı ve en az bir karakter içermelidir).
4. Kullanılan şifreler düzmetin olarak saklanmamalıdır. Bunun yerine şifreleme fonksiyonlarının çıktıları tutulmalıdır.
5. Şifre güncelleme işlemlerinde kullanıcının doğruluğu birden fazla parametre ile kontrol edilmelidir.
6. Session bilgisini sadece COOKIE yardımıyla değil, IP ve zaman kontrolleri ile de desteklemek gerekir.
7. Uygulama kullanıcısına hiç bir zaman hata rapor edilmemelidir. Bunun yerine hata kodları gösterilmeli ve bu kodlar yardımıyla debug işlemleri yapılmalıdır.

8. GET, POST, COOKIE v.s gibi parametreler her zaman güvensiz kabul edilmeli ve işlem görmeden önce doğruluğu kontrol edilmelidir.

9. Tüm girdi parametreleri server tarafında kontrol edilmelidir.

10. Tüm çıktı parametreleri kontrol edilmelidir.

11. Opensorce kodlar (özellikle JS modülleri) kontrol edilerek kullanılmalıdır.

12. Özellikle SQL injection`a karşı prepared statement/parameterized query/bind variables yöntemi kullanılmalıdır.

13. Uygulama`nın veritabanına bağlanan kullanıcısının gereksiz tüm hakları alınmalıdır.

14.Uygulama`nın veritabanı kullanıcısının sadece local erişimi olmalıdır.

15. DoS saldırısı barındıracak tüm formlara CAPTHCA eklenmelidir.

16. Özellikle finansal işlemler veya buna benzer kritik işlemler için bir formdan diğer forma veya formun kendisinin kaydedilmesi esnasında, gerçekten belirtilen formdan geldiğinin kontrol edilmesi için TOKEN eklenmeli ve kontrol edilmelidir. Özellikle CSRF saldırılarının önlemi için gereklidir.

17. Dosya kabul eden formlar için dosya boyut, tip kontrolleri server tarafında yapılmalıdır ve dosyaların saklandığı dizinlerden dış dizinlere çıkılmamalıdır.

Umarım denetim ve kod yazım esnasında bu başlıklara dikkat ediyoruzdur. Aksi durumda ciddi tehlikeler ile başbaşayız diyebilirim.