**Cookie Not Marked as Secure**

**Çerez Güvenli Olarak İşaretlenmedi**  **[1] :**

**çerezlerin güvenliğini sağlamak önemli bir konudur.**

**bir kimlik doğrulama çerezi düşünelim. Saldırgan bu çerezi yakalayabildiğinde, kullanıcının kimliğine bürünebilir.**

**HTTP protokolü kullanıldığında, trafik düz metin olarak gönderilir. Saldırganın trafiği görmesini/değiştirmesini sağlar (man-in-the-middle attack). HTTPS, HTTP'nin güvenli bir sürümüdür — uygulama katmanının verilerini korumak için SSL/TLS kullanır.**

**HTTPS kullanıldığında şu özellikler elde edilir: kimlik doğrulama, veri bütünlüğü ve gizlilik.**

**Bir kimlik doğrulama çerezi durumunu ele alalım. Bu çerezi çalmak, kullanıcının kimliğine bürünmekle eşdeğerdir. HTTP kullanıldığında, çerez düz metin olarak gönderilir. Bu, tarayıcı ve sunucu arasındaki iletişim kanalını gizlice dinleyen saldırgan için iyidir - çerezi alıp kullanıcının kimliğine bürünebilir.**

**Şimdi HTTP yerine HTTPS kullanıldığını varsayalım. HTTPS gizlilik sağlar. Bu nedenle saldırgan çerezi göremez. Sonuç, kimlik doğrulama çerezini gizlice dinlenmemesi için güvenli bir kanal üzerinden göndermektir.**

**Şu anda ortaya çıkabilecek soru şudur: HTTPS kullanabiliyorsak neden güvenli bir bayrağa ihtiyacımız var?**

**Bu soruyu cevaplamak için aşağıdaki senaryoyu ele alalım. Site, HTTP ve HTTPS üzerinden kullanılabilir. Ayrıca tarayıcı ile sunucu arasındaki iletişim kanalının ortasında bir saldırganın olduğunu varsayalım. HTTPS üzerinden gönderilen çerez dinlenemez.**

**Ancak saldırgan, sitenin HTTP üzerinden de erişilebilir olmasından faydalanabilir. Saldırgan, sitenin HTTP sürümünün bağlantısını kullanıcıya gönderebilir. Kullanıcı bağlantıyı tıklar ve HTTP isteği oluşturulur. HTTP trafiği düz metin olarak gönderildiğinden, saldırgan iletişim kanalını gizlice dinler ve kullanıcının kimlik doğrulama çerezini okur. Bu çerezin yalnızca HTTPS üzerinden gönderilmesine izin verebilir miyiz? Bu mümkün olsaydı, saldırganın hikayemizdeki kimlik doğrulama çerezini okumasını engellerdik. Bunun mümkün olduğu ve tam olarak bu amaç için güvenli bir bayrak kullanıldığı ortaya çıktı - güvenli bir bayrağa sahip çerez yalnızca bir HTTPS bağlantısı üzerinden gönderilecek.**

**Biraz önce tarayıcı ile sunucu arasındaki iletişim kanalını gizlice dinleyen bir saldırgandan çerezin nasıl korunacağı anlatılmıştı. Ancak, gizlice dinleme, çerezi kapmak için tek saldırı vektörü değildir.**

**Hikayeye kimlik doğrulama çerezi ile devam edelim ve uygulamada XSS (siteler arası komut dosyası çalıştırma) güvenlik açığı olduğunu varsayalım. Ardından saldırgan, kimlik doğrulama tanımlama bilgisini çalmak için XSS güvenlik açığından yararlanabilir. Bunun olmasını bir şekilde önleyebilir miyiz?**

**Bu sorunu çözmek için bir HttpOnly bayrağının kullanılabileceği ortaya çıktı. HttpOnly bayrağı kullanıldığında, XSS istismarı durumunda JavaScript bu kimlik doğrulama tanımlama bilgisini okuyamaz. Hedefe ulaştık gibi**

**Özet [2]:**

Invicti, güvenli olarak işaretlenmeyen ve HTTPS üzerinden aktarılan bir tanımlama bilgisi belirledi. Bu, çerezin trafiği başarıyla kesip şifresini çözebilen bir saldırgan tarafından veya başarılı bir ortadaki adam saldırısını takiben potansiyel olarak çalınabileceği anlamına gelir

**Başarılı Sömürü için Gerekli Beceriler**

Bu sorundan yararlanmak için saldırganın trafiği kesebilmesi gerekir. Bu genellikle web sunucusuna veya kurbanın ağına yerel erişim gerektirir. Saldırganların 2. katmanı anlaması, trafik için geçiş noktaları olarak sistemlere fiziksel erişime sahip olması veya kurban ile web sunucusu arasındaki bir sisteme yerel olarak erişim kazanmış olması gerekir.

**Robots.txt**

**GİRİŞ :**

**arama motorları kendi botları aracılığıyla web sitelerini düzenli olarak tarar ve tarama sonuçlarını arşivler. Ancak bazı durumlarda web sitenizin her kısmını botlar için görünür kılmak istemiyor olabilirsiniz. Botların görmesinin önüne geçmek istediğiniz bölümler içinse en iyi yöntem ise robots.txt dosyasını kullanmak olacaktır**

**Robots.txt Nedir [3]?**

**Web sitenizdeki hangi dosya ya da sekmelerin tarayıcı botlarına açık olduğunu seçmenize imkan tanıyan robots.txt dosyası, sitenizin kök dizininde yer alır. 'Robot hariç tutma standartları' çerçevesinde oluşturulan bu dosya, aslında düz bir metinden oluşur. Ancak dosya içeriğinde bir veya daha fazla komut bulunur. Her komutla birlikte farklı bir tarayıcının, ayrı bir dosyaya erişmesi engellenir.**

**Robots.txt içerisinde kullanılan allow ve disallow direktifleri site içerisindeki hangi sayfaların taramaya açılıp kapanacağının belirtildiği protokollerdir.**

**Allow’un türkçe karşılığı izin vermek ve Disallow’un karşılığı izin vermemektir.**

**Bu bilgiden hareketle Allow: protokol kulllanımı ile site içerisindeki herhangi bir dizini, URL yolunu tümüyle yada kısmen taramalara açabilirsiniz.**

**Aynı şekilde Disallow: kullanımı ile herhangi bir dizini, URL yolunu tümüyle yada kısmi olarak taramalara kapatabilirsiniz.**

**Peki reski nedir ?**robots.txt dosyasının varlığı kendi başına herhangi bir güvenlik açığı oluşturmaz. Ancak, genellikle bir sitenin içeriğinin kısıtlı veya özel alanlarını belirlemek için kullanılır. Bu nedenle, dosyadaki bilgiler, özellikle tanımlanan konumlardan bazıları sitenin başka bir yerinden bağlantılı değilse, bir saldırganın sitenin içeriğinin haritasını çıkarmasına yardımcı olabilir. Uygulama, bu alanlara erişimi korumak için robots.txt'ye güveniyorsa ve bunlar üzerinde uygun erişim denetimi uygulamıyorsa, bu ciddi bir güvenlik açığı oluşturur.

**KAYNAKLAR :**

**[1] :**

[**https://silesiasecuritylab.com/free-resources/**](https://silesiasecuritylab.com/free-resources/)

**Author: Dawid Czagan**

**Posted: August/10/2020**

**[2]:**

[**https://www.invicti.com/web-vulnerability-scanner/vulnerabilities/cookie-not-marked-as-secure/**](https://www.invicti.com/web-vulnerability-scanner/vulnerabilities/cookie-not-marked-as-secure/)

**[3]:**

**https://boosmart.com/robots-txt-nedir-nasil-kullanilir/#Robotstxt\_Dosyasi\_Nerede\_Barindirilir\_Bulunur**