Testing for Session Management Schema (Oturum Yönetimi Şeması için Test)

Summary (Özet)

Herhangi bir web tabanlı uygulamanın temel bileşenlerinden biri, onunla etkileşime giren bir kullanıcı için devleti kontrol ettiği ve koruduğu mekanizmadır. Bir web sitesinin veya hizmetinin her sayfası için sürekli kimlik doğrulamasını önlemek için, web uygulamaları önceden belirlenmiş bir zaman aralığı için kimlik bilgilerini depolamak ve doğrulamak için çeşitli mekanizmalar uygular. Bu mekanizmalar Oturum Yönetimi olarak bilinir.

Bu testte test cihazı, çerezlerin ve diğer oturum belirteçlerinin güvenli ve öngörülemeyen bir şekilde oluşturulduğunu kontrol etmek istiyor. Zayıf bir çerezi tahmin edebilen ve kurabilen bir saldırgan,
meşru kullanıcıların oturumlarını kolayca kaçırabilir.

Çerezler oturum yönetimini uygulamak için kullanılır ve RFC 2965'te ayrıntılı olarak açıklanmıştır. Özetle, bir kullanıcı bu kullanıcının eylemlerini ve kimliğini birden fazla istekle takip etmesi gereken bir uygulamaya eriştiğinde, sunucu tarafından bir çerez (veya çerezler) oluşturulur ve istemciye gönderilir. Müşteri daha sonra çerez süresi dolana veya yok edilene kadar aşağıdaki bağlantılarda çerezi sunucuya geri gönderir. Çerezde depolanan veriler, sunucuya kullanıcının kim olduğu, şimdiye kadar hangi eylemleri gerçekleştirdiği, tercihlerinin ne olduğu vb. hakkında büyük bir bilgi yelpazesi sağlayabilir. Bu nedenle HTTP gibi vatansız bir protokole devlet sağlar.

Tipik bir örnek, bir çevrimiçi alışveriş sepeti tarafından sağlanır. Bir kullanıcının oturumu boyunca, uygulama kimliğini, profilini, satın almayı seçtiği ürünleri, miktarı, bireysel fiyatları, indirimleri vb.
Takip etmelidir. Çerezler, bu bilgileri ileri geri depolamak ve aktarmak için etkili bir yoldur (diğer yöntemler URL parametreleri ve gizli alanlardır).

Sakladıkları verilerin önemi nedeniyle, çerezler bu nedenle uygulamanın genel güvenliğinde hayati öneme sahiptir. Çerezleri kurcalayabilmek, meşru kullanıcıların

oturumlarını kaçırmak, aktif bir
oturumda daha yüksek ayrıcalıklar elde etmek ve genel olarak uygulamanın işlemlerini yetkisiz bir şekilde etkilemekle sonuçlanabilir.

Bu testte test cihazı, müşterilere verilen çerezlerin meşru kullanıcıların oturumlarına ve uygulamanın kendisiyle müdahale etmeyi amaçlayan çok çeşitli saldırılara direnip direnemeyeceğini kontrol etmek zorundadır. Genel amaç, uygulama tarafından geçerli sayılacak ve bir tür yetkisiz erişim sağlayacak bir çerez oluşturabilmektir (oturum kaçırma, ayrıcalık esrar, ...).

Genellikle saldırı modelinin ana adımları şunlardır:

- **Çerez** toplama: yeterli sayıda çerez örneği toplama;
- **çerez ters mühendisliği** : çerez oluşturma algoritmasının analizi;
- **Kurabiye** manipülasyonu: saldırıyı gerçekleştirmek için geçerli bir kurabiyenin dövülmesi. Bu son adım, çerezin nasıl
  oluşturulduğuna (çerez kaba kuvvet saldırısı) bağlı olarak çok sayıda deneme gerektirebilir.

Bir başka saldırı paterni de bir kurabiyenin taşmasından oluşur. Kesinlikle konuşursak, bu saldırı farklı bir doğaya sahiptir, çünkü burada testçiler mükemmel bir şekilde geçerli bir çerezi yeniden
yaratmaya çalışmamaktadır. Bunun yerine, amaç bir bellek alanını taşmak, böylece uygulamanın doğru davranışına müdahale etmek ve muhtemelen kötü amaçlı kod enjekte etmek (ve uzaktan yürütme).

Test Objectives (Test Hedefleri)

- Aynı kullanıcı için ve mümkün olan yerlerde farklı kullanıcılar için oturum belirteçleri toplayın.
- Oturum oluşturma saldırılarını durdurmak için yeterli rastgeleliğin var olmasını analiz edin ve sağlayın.
- İmzalı olmayan çerezleri değiştirin ve manipüle edilebilecek bilgiler içerir.

How to Test(Nasıl Test Edilir)

Black-Box Testing and Examples (**Siyah-Kutu Test ve Örnekleri**)

Müşteri ve uygulama arasındaki tüm etkileşim en azından aşağıdaki kriterlere göre test edilmelidir:

- Hepsi var `Set-Cookie` Direktifler olarak etiketlenmiş `Secure` ?
- Şifrelenmemiş taşıma üzerinden herhangi bir Çerez operasyonu gerçekleşir mi?
- Kurabiye şifrelenmemiş ulaşım için zorlanabilir mi?
- Eğer öyleyse, uygulama güvenliği nasıl koruyor?
- Çerezler kalıcı mıdır?
- Ne `Expires` Zaman kalıcı çerezlerde kullanılır ve bunlar makul mü?
- Bu şekilde geçici yapılandırılması beklenen çerezler mi?
- HTTP/1.1 Ne `Cache-Control` Çerezleri korumak için ayarlar kullanılır mı?
- NeHDe HTTP/1.0 `Cache-Control` Çerezleri korumak için ayarlar kullanılır mı?

Cookie Collection (Çerez Koleksiyonu)

Kurcayı manipüle etmek için gereken ilk adım, uygulamanın çerezleri nasıl oluşturduğunu ve yönettiğini anlamaktır. Bu görev için, testçiler aşağıdaki soruları yanıtlamaya çalışmak zorundadır:

- Uygulama tarafından kaç çerez kullanılıyor?

  Uygulamayı sağla. Çerezler oluşturulduğu zaman not edin. Alınan çerezlerin bir listesini, onları (bölge-çereketli direktifle), geçerli oldukları etki alanı, değerlerini ve özelliklerini belirleyen sayfayı yapın.

- Uygulamanın hangi parçaları çerezi oluşturur veya değiştirir?

  Uygulamayı tarayarak, hangi çerezlerin sabit kaldığını ve hangilerinin değiştirildiğini öğrenin. Çerezleri hangi olayları değiştirir?

- Uygulamanın hangi bölümlerine erişilebilmesi ve kullanılması için bu çerezi gerektirir?

  Uygulamanın hangi bölümlerinin bir çereze ihtiyacı olduğunu öğrenin. Bir sayfaya erişin, ardından çerez olmadan veya değiştirilmiş bir değeri ile tekrar deneyin. Hangi çerezlerin kullanıldığı yerleri haritalamaya çalışın.

Her çerezi ilgili uygulama parçalarına ve ilgili bilgileri eşleştiren bir elektronik tablo bu fazın değerli bir çıktısı olabilir.

Session Analysis (Oturum Analizi)

Oturum tokenleri (Cookie, SessionID veya Hidden Field) kalitelerini güvenlik perspektifinden sağlamak için incelenmelidir. Rastgelelik, benzersizlik, istatistiksel ve kriptografik analize karşı direnç ve bilgi sızıntısı gibi kriterlere göre test edilmelidirler.

- Token Yapısı ve Bilgi Sızıntısı

İlk aşama, uygulama tarafından sağlanan bir Oturum Kimliğinin yapısını ve içeriğini incelemektir. Yaygın bir hata, jenerik bir değer vermek ve gerçek veri sunucusuna atıfta bulunmak yerine Token'e belirli verileri dahil etmektir.

Oturum Kimliği net metin ise, yapı ve ilgili veriler hemen belli olabilir. `192.168.100.1:owaspuser:password:15:58` . .

Parça veya tüm token kodlanmış veya haşlanmış gibi görünüyorsa, bariz karışıklıkları kontrol etmek için çeşitli tekniklerle karşılaştırılmalıdır. Örneğin dize `192.168.100.1:owaspuser:password:15:58` Hex, Base64'te ve MD5 hash olarak temsil edilir:

- Hex: `3139322E3136382E3130302E313A6F77617370757365723A70617373776F72643A31353A3538`
- Base64: `MTkyLjE2OC4xMDAuMTpvd2FzcHVzZXI6cGFzc3dvcmQ6MTU6NTg=`
- MD5: `01c2fc4f0a817afd8366689bd29dd40a`

Bu tür bir bulanıklık tespit ettikten sonra, orijinal verilere geri kodlanması mümkün olabilir. Bununla birlikte, çoğu durumda, bu pek olası değildir. Buna rağmen, kodlamayı mesajın biçiminden yerleştirmek yararlı olabilir. Ayrıca, hem format hem de bulanıklaştırma tekniği çıkarılabilirse, otomatik kaba kuvvet saldırıları tasarlanabilir.

Hibrit belirteçler, IP adresi veya Kullanıcı Kimliği gibi kodlanmış bir bölümle birlikte bilgi içerebilir. `owaspuser:192.168.100.1:a7656fafe94dae72b1e1487670148412` . .

Tek bir oturum belirtecini analiz ettikten sonra, temsili örnek incelenmelidir. Tokenlerin basit bir analizi derhal herhangi bir belirgin kalıbı ortaya çıkarmalıdır. Örneğin, 32 bit bir token 16 bit
statik veri ve 16 bit değişken veri içerebilir. Bu, ilk 16 bitlerin kullanıcının sabit bir özelliğini temsil ettiğini gösterebilir - örneğin kullanıcı adı veya IP adresi. İkinci 16

parça parça düzenli bir oranda
artıyorsa, token nesline sıralı veya hatta zamana dayalı bir unsura işaret edebilir. Örnekleri görün.

Tokenlerin statik elemanları tanımlanırsa, bir seferde potansiyel bir giriş elemanını değiştirerek daha fazla numune toplanmalıdır. Örneğin, farklı bir kullanıcı hesabı üzerinden veya farklı bir IP adresinden girişimlerde oturum açmaya işaret eden daha önceki statik kısımda bir
varyans verebilir.

Aşağıdaki alanlar tek ve çoklu Oturumlu kimlik yapısı testi sırasında ele alınmalıdır:

- Oturum kimliğinin hangi bölümleri statiktir?
- Session ID'de hangi net metin gizli bilgiler saklanıyor? E.g. kullanıcı adları/UID, IP adresleri
- Gizli bilgilerin kolayca çözüldüğü nedir?
- Oturum Kimliği'nin yapısından hangi bilgiler çıkarılabilir?
- Oturum Kimliğinin hangi bölümleri aynı gün için statiktir?
- Oturum Kimliğinde bir bütün olarak veya bireysel bölümlerde hangi bariz kalıplar vardır?

Session ID Predictability and Randomness (Oturum ID Öngörülebilirliği ve Randoness)

Oturum Kimliğinin değişken alanlarının (varsa) analizi, tanınabilir veya öngörülebilir herhangi bir kalıbın varlığını belirlemek için yapılmalıdır. Bu analizler, Oturum Kimliği içeriğindeki herhangi bir
kalıptan çıkarmak için manuel olarak ve ısmarlama veya OTS istatistiksel veya kriptonalitik araçlarla gerçekleştirilebilir. Manuel kontroller, aynı giriş koşulları için verilen Oturum Kimliklerinin
karşılaştırmalarını içermelidir - örneğin, aynı kullanıcı adı, şifre ve IP adresi.

Zaman da kontrol edilmesi gereken önemli bir faktördür. Örnekleri aynı zaman penceresinde toplamak ve bu değişken sabit tutmak için yüksek sayıda eşzamanlı bağlantı yapılmalıdır. 50ms veya daha az bir miktar bile çok daha kaba olabilir ve bu şekilde alınan bir örnek, aksi
takdirde kaçırılacak zaman tabanlı bileşenleri ortaya çıkarabilir.

Doğada artan olup olmadıklarını belirlemek için Değişken elementler zamanla analiz edilmelidir. Artan olduklarında, mutlak veya geçmiş zaman ile ilgili kalıplar araştırılmalıdır. Birçok sistem zamanı sahte rastgele elementleri için tohum olarak kullanır. Kalıpların görünüşte rastgele olduğu durumlarda, tek yönlü zaman veya diğer çevresel varyasyonlar bir olasılık olarak düşünülmelidir. Tipik olarak, bir kriptografik hashin sonucu ondalık veya altı eksendemimal bir sayıdır, bu nedenle tanımlanabilir olmalıdır.

Oturum Kimliği dizilerini, kalıplarını veya döngülerini analiz ederken, statik elemanlar ve istemci bağımlılıkları, uygulamanın yapısına ve işlevine katkıda bulunan unsurlar olarak düşünülmelidir.

- Session Kimlikleri doğada rastgele mi? Ortaya çıkan değerler yeniden üretilebilir mi?
- Aynı giriş koşulları sonraki bir çalışmada aynı kimliği üretir mi?
- Oturum Kimlikleri istatistiksel veya kriptanalizliğe karşı hassas mı?
- Oturum Kimliklerinin hangi unsurları zaman bağlantılıdır?
- Oturum Kimliklerinin hangi bölümleri tahmin edilebilir?
- Bir sonraki kimlik, nesil algoritması ve önceki kimlikler hakkında tam bilgi verildiğinde çıkarılabilir mi?

Cookie Reverse Engineering (Çerez Ters Mühendisliği)

Şimdi testçi kurabiyeleri numaralandırdığına ve kullanımları hakkında genel bir fikri olduğuna göre, ilginç görünen çerezlere daha derin bir bakmanın zamanı geldi. Test cihazı hangi çerezlerle ilgileniyor? Güvenli bir oturum yönetimi yöntemi sağlamak için bir çerez, her biri çerezi
farklı bir saldırı sınıfından korumayı amaçlayan çeşitli özellikleri birleştirmelidir.

Bu özellikler aşağıda özetlenmiştir:

1. Öngörülemezlik: Bir çerez, tahmin edilmesi zor bir miktar veri içermemelidir. Geçerli bir çerez oluşturmak ne kadar zor olursa, meşru kullanıcının oturumuna girmek daha zordur. Bir saldırgan meşru bir kullanıcının aktif bir oturumunda kullanılan çerezi tahmin edebilirse, bu kullanıcıyı (oturum kaçırma) tamamen taklit edebilecektir. Bir çerezi öngörülemez hale getirmek için rastgele değerler veya kriptografi kullanılabilir.

2. Tamper direnci: bir çerez, kötü niyetli modifikasyon girişimlerine direnmelidir. Test cihazı benzer bir kurabiye alırsa `IsAdmin=No`, uygulama iki kez kontrol sağlamadığı sürece, idari haklar elde etmek için değiştirmek önemsizdir (örneğin, çereze değerinde şifreli bir hash eklenir)
3. Sona Erme: kritik bir çerez yalnızca uygun bir süre için geçerli olmalıdır ve tekrar oynatılma riskini önlemek için daha sonra diskten veya bellekten silinmelidir. Bu, oturumlar arasında hatırlanması gereken kritik olmayan verileri saklayan çerezler için geçerli değildir (örneğin, site görünümü ve hissetme).
4. `Secure` Bayrak: Oturumun bütünlüğü için değeri kritik olan bir çerez, bu bayrağın sadece şifreli bir kanalda dinlemeyi caydırmak için iletilmesine izin vermek için etkinleştirilmelidir.

Buradaki yaklaşım, bir çerezin yeterli sayıda örneğini toplamak ve değerlerinde desenler aramaya başlamaktır. "Yeterli"nin tam anlamı, çerez üretim yönteminin kırılması çok kolaysa, test cihazının bazı matematiksel analizlere (örneğin, chi-kareler, çekiciler) ile devam etmesi gerekiyorsa, birkaç binlere kadar değişebilir. Daha fazla bilgi için daha sonra bakın).

Başvurunun iş akışına özellikle dikkat etmek önemlidir, çünkü bir oturumun durumu toplanan çerezler üzerinde ağır bir etkiye sahip olabilir. Doğrulanmadan önce toplanan bir çerez, kimlik doğrulamadan sonra elde edilen bir çerezden çok farklı olabilir.

Dikkate alınması gereken bir diğer husus da zaman. Her zaman bir çerezin elde edildiği tam zamanı kaydedin, çerezin değerinde zamanın bir rol oynama olasılığı olduğunda (sunucu, çerez değerinin bir parçası olarak bir zaman damgasını kullanabilir). Kaydedilen zaman, yerel saat veya sunucunun zaman damgası HTTP yanıtına (veya her ikisine) dahil edilebilir.

Toplanan değerleri analiz ederken, test cihazı çerez değerini etkileyebilecek tüm değişkenleri anlamaya çalışmalı ve o sırada bunları değiştirmeye çalışmalıdır. Aynı çerezin sunucu değiştirilmiş sürümlerine geçmek, uygulamanın çerezi nasıl okuduğunu ve işlediğini anlamada çok yardımcı olabilir.

Bu aşamada gerçekleştirilecek kontrollerin örnekleri şunları içerir:

- Çerezde hangi karakter seti kullanılır? Kurabiye sayısal bir değere sahip mi? Alfanömary mü? Altı eksenli mi? Tester, beklenen kreşe ait olmayan bir çerez karakterine eklerse ne olur?

- Kurs, farklı bilgi parçaları taşıyan farklı alt kısımlardan oluşuyor mu? Farklı parçalar nasıl ayrılır? Hangi sınırlamalarla? Kurabiyenin bazı bölümleri daha yüksek bir varlığa sahip olabilir, diğerleri sabit olabilir, diğerleri sadece sınırlı bir değerler kümesi varsayabilir. Çerezi temel bileşenlerine ayırmak ilk ve temel adımdır.

Spotu yapılandırılması kolay bir çerez örneği şunlardır:

`ID=5a0acfc7ffeb919:CR=1:TM=1120514521:LM=1120514521:S=j3am5KzC4v01ba3q`

Bu örnek, farklı veri türlerini taşıyan 5 farklı alanı göstermektedir:

- ID – hexadecimal
- CR – küçük biracı
- TM ve LM - büyük payet. (Ve merakla aynı değeri taşıyorlar. Bunlardan birini değiştirmek için ne olduğunu görmek için değer)
- S – alfanumeric

Hiçbir sınırlama kullanılmadığında bile, yeterli numuneye sahip olmak yapıyı anlamaya yardımcı olabilir.

Brute Force Attacks (Kaba Kuvvet Saldırıları)

Kaba kuvvet atakları kaçınılmaz olarak öngörülebilirlik ve rastgelelik ile ilgili sorulardan kaynaklanmaktadır. Oturum Kimlikleri içindeki varyans, başvuru oturumu süresi ve zaman aşımları ile birlikte düşünülmelidir. Oturum Kimlikleri içindeki varyasyon nispeten azsa ve Oturum Kimliği geçerliliği uzunsa, başarılı bir kaba kuvvet saldırısı olasılığı çok daha yüksektir.

Uzun bir Oturum Kimliği (veya daha doğrusu büyük bir varyansla) ve daha kısa bir geçerlilik süresi, kaba bir kuvvet saldırısında başarılı olmayı çok daha zor hale getirecektir.

- Olası tüm Oturum Kimliklerine kaba kuvvet saldırısı ne kadar sürer?
- Oturum Kimliği alanı kaba kuvvetlendirmeyi önleyecek kadar büyük mü? Örneğin, anahtarın uzunluğu geçerli yaşam süresine kıyasla yeterli midir?
- Farklı Oturum Kimlikleri ile bağlantı girişimleri arasındaki gecikmeler bu saldırı riskini azaltır mı?

Gray-Box Testing and Example (Gri-Kutu Test ve Örnek)

Test cihazının oturum yönetimi şeması uygulamasına erişimi varsa aşağıdakileri kontrol edebilirler:

- Rastgele Oturumlu Jeton

  Müşteriye verilen Oturum Kimliği veya Çerez kolayca öngörülebilir olmamalıdır (müşteri IP adresi gibi öngörülebilir değişkenlere dayalı doğrusal algoritmalar kullanmayın). 256 bitlik anahtar uzunluğuna sahip kriptografik algoritmaların kullanımı teşvik edilir (AES gibi).

- Token uzunluğu

  Oturum kimliği en az 50 karakter uzunluğunda olacaktır.

- Oturum Zamanını Geçirme

  Oturum tokeni tanımlanmış bir zaman aşımına sahip olmalıdır (uygulama tarafından yönetilen verilerin kritikliğine bağlıdır)

- Çerez yapılandırması:
  - Kalıcı olmayan: sadece RAM hafızası
  - Güvenli (sadece HTTPS kanalında ayarlanır): `Set-Cookie: cookie=data; path=/; domain=.aaa.it; secure`
  - Sadece HTTP (bir senaryo tarafından okunamaz): `Set-Cookie: cookie=data; path=/; domain=.aaa.it; HttpOnly`

Burada daha fazla bilgi: Çerez özellikleri için test

Tools (Araçlar)

- OWASP Zed Saldırı Proxy Projesi (ZAP) - bir oturum token analiz mekanizmasına sahiptir.
- Burp Sequencer
- YEHG'nin JHijack'i

References (Referanslar)

Whitepapers (Beyaz kağıtlar)

- RFC 2965 “HTTP Devlet Yönetimi Mekanizması”

- RFC 1750 “Güvenlik için Rastgelelik Önerileri”
- Michal Zalewski: “Garip Çekiciler ve TCP/IP Diziliş Numarası Analizi” (2001)
- Michal Zalewski: “Garip Çekiciler ve TCP / IP Dizi Numarası Analizi – Bir Yıl Sonra” (2002)
- Korelasyon Katsayısı
- ENT
- DMA[2005-0614a] - 'Küresel Hauri ViRobot Sunucusu çerezi taşması'
- Gunter Ollmann: “Web Merkezli Oturum Yönetimi”
- OWASP Kod İnceleme Kılavuzu