Testing for Improper Error Handling (Uygunsuz Hata İşleme Testi)

Summary (Özet)

Tüm uygulama türleri (web uygulamaları, web sunucuları, veritabanları vb.) çeşitli nedenlerle hata üretecektir. Geliştiriciler genellikle bu hataları ele almayı görmezden gelir veya bir kullanıcının kasıtlı olarak bir hatayı tetiklemeye çalışacağı fikrini zorlarlar (*örneğin,* bir bütünleştiğinin beklendiği bir dize gönderir). Geliştirici sadece mutlu yolu düşündüklerinde, kodun alabileceği ancak kaldıramadığı diğer tüm olası kullanıcı girişlerini unuturlar.

Hatalar bazen şöyledir:

- Istif izleri,
- ağ zaman aşımları,
- Giriş uyumsuzluğu,
- ve hafıza çöplükleri.

Uygunsuz hata işleme saldırganların şunları yapmasına izin verebilir:

- Dahili olarak kullanılan API'leri anlayın.
- Kullanılan iç sistemler ve çerçeveler hakkında fikir edinerekbirbirleriyle bütünleşen çeşitli hizmetleri haritalar, bu da zincirlemesaldırmak için kapıları açar.
- Kullanılan uygulamaların versiyonlarını ve türlerini toplayın.
- Sistemi, sistemi bir çıkmaza veya çalıştıran motora panik sinyali gönderen bir ele alınmayan istisnaya zorlayarak sistemi yapar.
- Kontroller, belirli bir istisnanın mutlu yolun etrafında belirtilen mantıkla kısıtlanmadığı yerlerde atlar.

Test Objectives (Test Hedefleri)

- Mevcut hata çıktısını belirleyin.
- Geri dönen farklı çıktıyı analiz edin.

How To Test (Nasıl Test Edilir)

Hatalar genellikle tanılama verilerini ve kullanıcının eldeki sorunu anlamasına yardımcı olabilecek veya geliştiricinin bu hatayı reddetmesi için tanılama verilerini ve mesajları sağladıkları için iyi huylu olarak görülür.

Beklenmedik veriler göndermeye çalışarak veya sistemi belirli durumlara ve senaryolara zorlayarak, sistem veya uygulama, geliştiriciler tüm olası hataları kapatmadıkça ve belirli bir özel mesajı iade etmedikçe, çoğu zaman dahili olarak neler olup bittiğine dair biraz verecektir.

Web Servers (Web Sunucuları)

Tüm web uygulamaları, entegre bir veya tam bir tane olsun, bir web sunucusunda çalışır. Web uygulamaları HTTP isteklerini ele almalı ve ayrıştırmalı ve bunun için bir web sunucusu her zaman yığının bir parçasıdır. En ünlü web sunucularından bazıları NGINX, Apache ve IIS'tir.

Web sunucuları bilinen hata mesajları ve formatlara sahiptir. Eğer biri nasıl göründüklerine aşina değilse, onları aramak örnekler verir. Başka bir yol, belgelerine bakmak veya yerel olarak bir sunucu kurmak ve web sunucusunun kullandığı sayfalardan geçerek hataları keşfetmek olacaktır.

Hata mesajlarını tetiklemek için, bir test cihazının şunları yapması gerekir:

- Bulunmayacak rastgele dosya ve klasörleri arayın (404).
- Var olan klasörleri talep etmeye çalışın ve sunucu davranışını (403s, boş sayfa veya dizin listeleme).
- Kırılan bir istek göndermeyi deneyin HTTP RFC. . Bir örnek, çok büyük bir yol göndermek, başlıklar biçimini kırmak veya HTTP sürümünü değiştirmek olacaktır.
    - Hatalar uygulama düzeyinde ele alınsa bile, HTTP RFC'yi kırmak, entegre web sunucusunu kendisi gösterebilir, çünkü isteği işlemek zorundadır ve geliştiriciler bu hataları geçersiz kılmayı unutur.

Applications (Uygulamaları)

Uygulamalar, çok çeşitli hata mesajlarını serbest bırakmak için en duyarlı olanlardır: yığın izleri, bellek çöplükleri, yanlış kullanım istisnaları ve genel hatalar. Bu, uygulamaların çoğu zaman özel olarak inşa edildiği ve geliştiricilerin olası tüm hata durumlarını gözlemlemeleri ve ele almaları (veya küresel bir hata yakalama mekanizmasına sahip olmaları) ve bu hataların diğer hizmetlerle entegrasyonlardan ortaya çıkması nedeniyle olur.

Bir uygulama yapmak için bu hataları atın, bir test cihazı gerekir:

1. Uygulamanın veri beklediği olası giriş noktalarını belirleyin.
2. Beklenen giriş türünü (strings, Integrs, JSON, XML, vb.) analiz edin.
3. Daha odaklanmış bir test senaryosuna sahip olmak için önceki adımlara göre her giriş noktasını bulanıklaştırın.
    - Her girişi mümkün olan tüm enjeksiyonlarla bulaştırmak, sınırsız test süreniz olmadıkça en iyi çözüm değildir ve uygulama bu kadar girdiyi kaldırabilir.
    - Bulanıklık bir seçenek değilse, belirli bir parser kırmak için en yüksek şansa sahip uygulanabilir girdileri *seçin (örneğin,* bir JSON gövdesi için kapanış braketi, sadece birkaç karakterin beklendiği büyük bir metin, sunucular ve giriş doğrulama kontrolleri tarafından ayrıştırılabilecek parametrelerle CLRF enjeksiyonu, dosya isimleri için geçerli olmayan özel karakterler vb.).
    - Jargon verileriyle dolup taşlama, bazen tercümanlar geliştiricinin istisna işlemesinin dışında kırılacağı için her tür için çalıştırılmalıdır.
4. Hata mesajıyla yanıt veren hizmeti anlayın ve bu hizmetten daha fazla bilgi veya hata ayrıntısı ortaya çıkarmak için daha rafine bir füzz listesi yapmaya çalışın (bir veritabanı, bağımsız bir hizmet olabilir vb.).

Hata mesajları bazen sistemlerin haritalandırılmasında, özellikle bir mikro hizmet mimarisi altında ana zayıflıktır. Hizmetler hataları genel ve tek tip bir şekilde ele alacak şekilde düzgün bir şekilde
ayarlanmazsa, hata mesajları bir test cihazının hangi hizmetin hangi hizmeti talep ettiğini belirlemesine ve hizmet başına daha odaklı bir saldırıya izin vermesini sağlar.

> Testçinin yanıt türü için uyanık bir göz tutması gerekir. Bazen hatalar bir hata gövdesi ile başarı olarak iade edilir, hatayı 302'de gizler veya sadece bu hatayı temsil etmenin özel bir yoluna sahip olarak.

Remediation (Düzeltme)

Düzeltme için Proaktif Kontroller C10'a ve Hata İşleme Hile Sayfasına göz atın.

Playgrounds (Oyun Alanları)

- Juice Shop - Error Handling

References (Referanslar)

- WSTG: Appendix C - Fuzz Vectors
- Proactive Controls C10: Handle All Errors and Exceptions
- ASVS v4.1 v7.4: Error handling
- CWE 728 - Improper Error Handling
- Cheat Sheet Series: Error Handling