Identify Application Entry Points (Uygulama Giriş Noktalarını Belirleme)

Summary (Özet)

Uygulamayı ve saldırı yüzeyini numaralandırmak, test cihazının olası zayıflık alanlarını belirlemesine izin verdiği için, herhangi bir kapsamlı test yapılmadan önce önemli bir öncüdür. Bu bölüm, numaralandırma ve haritalama tamamlandıktan sonra araştırılması gereken uygulama içindeki alanların belirlenmesine ve haritalanmasına yardımcı olmayı amaçlamaktadır.

Test Objectives (Test Hedefleri)

- İstek ve yanıt analizi yoluyla olası giriş ve enjeksiyon noktalarını belirleyin.

How to Test (Nasıl Test Edilir)

Herhangi bir test başlamadan önce, test cihazı her zaman uygulamayı ve kullanıcı ve tarayıcının onunla nasıl iletişim kurduğunu iyi anlamalıdır. Test cihazı başvuruda yürürken, uygulamaya aktarılan her parametre ve form alanının yanı sıra tüm HTTP isteklerine dikkat etmelidirler. GET isteklerinin ne zaman kullanıldığına ve PAYLAŞIM taleplerinin uygulamaya parametreleri geçmek için kullanıldığı zamana özel dikkat göstermelidir. Ayrıca, RESTful hizmetleri için diğer yöntemlerin ne zaman kullanıldığına da dikkat etmeleri gerekir.

PSN talebi gibi talepler gövdesinde gönderilen parametreleri görmek için testçinin, ele geçiren bir vekil (bakım toolsaraçları) gibi bir araç kullanmak isteyebileceğini unutmayın. POST talebinde,
test cihazı uygulamaya aktarılan herhangi bir gizli form alanını da özel olarak not etmelidir, çünkü bunlar genellikle devlet bilgileri, öğelerin miktarı, öğelerin fiyatı, geliştiricinin kimsenin görmesini
veya değişmesini amaçlamadığı hassas bilgiler içerir.

Yazarın deneyiminde, bu test aşaması için bir vekil ve elektronik tablo kullanmak çok yararlı olmuştur. Vekil, test cihazı ile uygulama arasındaki her istek ve yanıtı

araştırırken takip edecektir. Ek olarak, bu noktada, testçiler genellikle her talebi ve yanıtı hapseder, böylece uygulamaya ve iade edilenlere tam olarak her başlığı, parametreyi vb. Görebilirler. Bu, özellikle büyük interaktif sitelerde (bir bankacılık başvurusunu düşünün) zamanlarda oldukça sıkıcı olabilir. Bununla birlikte, deneyim ne aranacağını gösterecektir ve bu aşama önemli ölçüde azaltılabilir.

Test cihazı uygulamadan geçerken, URL'deki ilginç parametreleri, özel başlıkları veya isteklerin / yanıtların gövdesine dikkat etmeli ve bunları bir elektronik tabloda kaydetmelidir. Elektronik tablo, istenen sayfayı (gelecek referans için proxy'den istek numarasını eklemek de iyi olabilir), ilginç parametreler, istek türünü (GET, POST, vb.), erişim doğrulanmış / okunmamışsa, TLS'nin çok adımlı bir işlemin parçasıysa, WebSckers kullanılmışsa ve diğer ilgili notları içermelidir. Uygulamanın her alanını haritalandırdıktan sonra, uygulamadan geçebilir ve tespit ettikleri alanların her birini test edebilir ve neyin işe yarayıp neyin işe yaramadığı ve neyin işe yaramadığına dair notlar verebilirler. Bu kılavuzun geri kalanı, bu ilgi alanlarının her birinin nasıl test edileceğini belirleyecektir, ancak bu bölüm gerçek testlerden herhangi biri başlamadan önce yapılmalıdır.

Aşağıda tüm talepler ve yanıtlar için bazı ilgi noktaları bulunmaktadır. Talepler bölümünde, GET ve POST yöntemlerine odaklanın, çünkü bunlar taleplerin çoğunluğu gibi görünür. PUT ve DELETE gibi diğer yöntemlerin kullanılabileceğini unutmayın. Çoğu zaman, bu daha nadir istekler, izin verilirse, güvenlik açıklarını ortaya çıkarabilir. Bu kılavuzda bu HTTP yöntemlerini test etmek için özel bir bölüm bulunmaktadır.

Talepler

- GET'lerin nerede kullanıldığını ve POST'lerin nerede kullanıldığını belirleyin.
- Bir POST talebinde kullanılan tüm parametreleri belirleyin (bunlar talebin gövdesindedir).
- POST talebinde, herhangi bir gizli parametreye özel dikkat edin. Bir POSTA gönderildiğinde, HTTP mesajının gövdesine uygulamaya gönderilir. Bunlar genellikle bir proxy veya HTML kaynak kodunu görüntüleme kullanılmadıkça görülmez. Buna ek olarak, gösterilen bir sonraki sayfa, verileri ve erişim seviyesi, gizli parametrenin (s)değerine bağlı olarak farklı olabilir.
- Bir GET isteğinde (yani URL), özellikle sorgu dizisinde (genellikle bir tane sonra) kullanılan tüm parametreleri belirleyin. Mark samarda samardır.

- Sorgi ipinin tüm parametrelerini belirleyin. Bunlar genellikle çift formattadır, örneğin `foo=bar`. . Ayrıca, birçok parametrenin bir ile ayrılmış gibi bir sorgu ipinde olabileceğini unutmayın. `&`, `\~`, `:`, ya da başka bir özel karakter ya da kodlama.
- Bir dizede veya bir POST talebinde birden fazla parametreyi belirlemek söz konusu olduğunda özel bir not, saldırıları gerçekleştirmek için parametrelerin bir kısmının veya tamamının gerekli
  olacağıdır. Test cihazının tüm parametreleri tanımlaması (ortadan veya şifrelenmiş olsa bile) ve hangilerinin uygulama tarafından işlendiğini belirlemelidir. Kılavuzun sonraki bölümleri bu parametrelerin nasıl test edileceğini belirleyecektir. Bu noktada, her birinin tespit edildiğinden emin olun.
- Ayrıca, tipik olarak görülmeyen herhangi bir ek veya özel tip başlıklara dikkat edin (örneğin `debug: false`).

Yanıtlar

- Yeni çerezlerin nerede ayarlandığını belirleyin (`Set-Cookie` Başlık), değiştirilmiş veya eklenir.
- Herhangi bir yönlendirmenin (3xx HTTP durum kodu), 400 durum kodunun, özellikle de 403 Yasak ve normal yanıtlar sırasında 500 dahili sunucu hatasının (yani, değiştirilmemiş istekler) nerede olduğunu belirleyin.
- Ayrıca ilginç başlıkların nerede kullanıldığını da unutmayın. Örneğin, `Server: BIG-IP` Sitenin yük dengeli olduğunu gösterir. Bu nedenle, bir site yük dengeli ve bir sunucu yanlış yapılandırılırsa, test cihazının kullanılan yük dengeleme türüne bağlı olarak savunmasız sunucuya erişmek için birden fazla istekte bulunması gerekebilir.

Black-Box Testing (Siyah-Box Testi)

Uygulama Giriş Noktaları için Test

Aşağıdakiler, başvuru giriş noktalarının nasıl kontrol edileceğine dair iki örnektir.

Örnek 1

Bu örnek, bir çevrimiçi alışveriş uygulamasından bir öğe satın alacak bir GET talebini göstermektedir.

```
GET /shoppingApp/buyme.asp?CUSTOMERID=100&ITEM=z101a&PRICE=62.5
0&IP=x.x.x.x HTTP/1.1
Host: x.x.x.x
Cookie: SESSIONID=Z29vZCBqb2IgcGFkYXdhIG15IHVzZXJuYW1lIGIzIGZvbyB
hbmQgcGFzc3dvcmQgaXMgYmFy
```

> Burada test cihazı, CUSTOMERID, ITEM, PRICE, IP ve Çerez (sadece
> kodlanmış parametreler veya oturum durumu için kullanılabilen) gibi
> talebin tüm parametrelerini not edecektir.

Örnek 2

Bu örnek, sizi bir uygulamaya kaydedecek bir POST talebini göstermektedir.

```
POST /KevinNotSoGoodApp/authenticate.asp?service=login HTTP/1.1
Host: x.x.x.x
Cookie: SESSIONID=dGhpcyBpcyBhIGJhZCBhcHAgdGhhdCBzZXRzIHByZWR
pY3RhYmxlIGNvb2tpZXMgYW5kIG1pbmUgaXMgMTIzNA==;CustomCookie=0
0my00trusted00ip00is00x.x.x.x00

user=admin&pass=pass123&debug=true&fromtrustIP=true
```

> Bu örnekte, test cihazı daha önce olduğu gibi tüm parametreleri not
> eder, ancak parametrelerin çoğu URL'de değil, isteğin gövdesinde geçer.
> Ayrıca, özel bir HTTP başlığı olduğunu unutmayın (CustomCookie) kullanılmaktan sonra.

Gray-Box Testing (Gri-Boks Testi)

Bir gri kutu metodolojisi aracılığıyla uygulama giriş noktaları için test, yukarıda bir ilave ile tanımlanan her şeyden oluşacaktır. Uygulamanın veri aldığı ve işlediği harici kaynakların bulunduğu durumlarda (SNMP tuzakları, sislog mesajları, SMTP veya diğer sunuculardan SAAP mesajları gibi) uygulama geliştiricileri ile bir toplantı, kullanıcı girdisini kabul edecek veya bekleyecek herhangi bir işlevi ve nasıl biçimlendirildiklerini tanımlayabilir. Örneğin, geliştirici, uygulamanın kabul edeceği doğru bir SOAP talebinin nasıl formüle edileceğini ve web hizmetinin nerede olduğunu anlamaya yardımcı olabilir (web hizmeti veya başka herhangi bir işlev kara kutu testi sırasında zaten tanımlanmadıysa).

OWASP Attack Surface Detector (OWASP Saldırı Yüzey Dedektörü)

Saldırı Yüzey Dedektörü (ASD) aracı kaynak kodunu araştırır ve bir web uygulamasının uç noktalarını, bu uç noktaların kabul ettiği parametreleri ve bu parametrelerin veri türünü ortaya çıkarır. Bu, bir örümceğin bulamayacağı uç noktaları veya istemci tarafı kodunda tamamen kullanılmamış isteğe bağlı parametreleri içerir. Ayrıca, bir uygulamanın iki versiyonu arasındaki saldırı yüzeyindeki değişiklikleri hesaplama yeteneğine sahiptir.

Attack Surface Detector hem ZAP hem de Burp Suite için bir eklenti olarak mevcuttur ve bir komut satırı aracı da mevcuttur. Komut çizgisi aracı, saldırı yüzeyini daha sonra ZAP ve Burp Suite eklentisi tarafından kullanılabilen bir JSON çıktısı olarak dışa aktarır. Bu, kaynak kodunun penetrasyon test cihazına doğrudan verilmediği durumlar için yararlıdır. Örneğin, penetrasyon test cihazı, kaynak kodunun kendisini sağlamak istemeyen bir müşteriden json çıktı dosyasını alabilir.

Nasıl kullanılır

CLI kavanoz dosyası https://github.com/secdec/attack-surface-detector-cli/releases adresinden indirilebilir.

Hedef web uygulamasının kaynak kodundan uç noktaları tanımlamak için ASD için aşağıdaki komutu çalıştırabilirsiniz.

```
java -jar attack-surface-detector-cli-1.3.5.jar <source-code-path> [flags]
```

İşte OWASP RailsGoat'a karşı komutu çalıştırmanın bir örneği.

```
$ java -jar attack-surface-detector-cli-1.3.5.jar railsgoat/
Beginning endpoint detection for '<...>/railsgoat' with 1 framework types
Using framework=RAILS
[0] GET: /login (0 variants): PARAMETERS={url=name=url, paramType=QUER
Y_STRING, dataType=STRING}; FILE=/app/controllers/sessions_contro
ller.rb (lines '6'-'9')
[1] GET: /logout (0 variants): PARAMETERS={}; FILE=/app/controllers/sessions
_controller.rb (lines '33'-'37')
[2] POST: /forgot_password (0 variants): PARAMETERS={email=name=email,
paramType=QUERY_STRING, dataType=STRING}; FILE=/app/controllers/
password_resets_controller.rb (lines '29'-'38')
[3] GET: /password_resets (0 variants): PARAMETERS={token=name=token, p
aramType=QUERY_STRING, dataType=STRING}; FILE=/app/controllers/p
assword_resets_controller.rb (lines '19'-'27')
[4] POST: /password_resets (0 variants): PARAMETERS={password=name=pa
ssword, paramType=QUERY_STRING, dataType=STRING, user=name=user, p
aramType=QUERY_STRING, dataType=STRING, confirm_password=name=co
nfirm_password, paramType=QUERY_STRING, dataType=STRING}; FILE=/ap
p/controllers/password_resets_controller.rb (lines '5'-'17')
[5] GET: /sessions/new (0 variants): PARAMETERS={url=name=url, paramTyp
e=QUERY_STRING, dataType=STRING}; FILE=/app/controllers/sessions_contr
oller.rb (lines '6'-'9')
[6] POST: /sessions (0 variants): PARAMETERS={password=name=password,
paramType=QUERY_STRING, dataType=STRING, user_id=name=user_id, para
mType=SESSION, dataType=STRING, remember_me=name=remember_me, p
aramType=QUERY_STRING, dataType=STRING, url=name=url, paramType=Q
UERY_STRING, dataType=STRING, email=name=email, paramType=QUERY_S
TRING, dataType=STRING}; FILE=/app/controllers/sessions_controller.rb (line
s '11'-'31')
[7] DELETE: /sessions/{id} (0 variants): PARAMETERS={}; FILE=/app/controlle
rs/sessions_controller.rb (lines '33'-'37')
[8] GET: /users (0 variants): PARAMETERS={}; FILE=/app/controllers/api/v1/us
ers_controller.rb (lines '9'-'11')
[9] GET: /users/{id} (0 variants): PARAMETERS={}; FILE=/app/controllers/api/
v1/users_controller.rb (lines '13'-'15')
```

```
... snipped ...
[38] GET: /api/v1/mobile/{id} (0 variants): PARAMETERS={id=name=id, param
Type=QUERY_STRING, dataType=STRING, class=name=class, paramType=Q
UERY_STRING, dataType=STRING}; FILE=/app/controllers/api/v1/mobile_contr
oller.rb (lines '8'-'13')
[39] GET: / (0 variants): PARAMETERS={url=name=url, paramType=QUERY_S
TRING, dataType=STRING}; FILE=/app/controllers/sessions_controller.rb (line
s '6'-'9')
Generated 40 distinct endpoints with 0 variants for a total of 40 endpoints
Successfully validated serialization for these endpoints
0 endpoints were missing code start line
0 endpoints were missing code end line
0 endpoints had the same code start and end line
Generated 36 distinct parameters
Generated 36 total parameters
- 36/36 have their data type
- 0/36 have a list of accepted values
- 36/36 have their parameter type
--- QUERY_STRING: 35
--- SESSION: 1
Finished endpoint detection for '<...>/railsgoat'
----------
-- DONE --
0 projects had duplicate endpoints
Generated 40 distinct endpoints
Generated 40 total endpoints
Generated 36 distinct parameters
Generated 36 total parameters
1/1 projects had endpoints generated
To enable logging include the -debug argument
```

Ayrıca bir JSON çıkış dosyası oluşturabilirsiniz. `-json` eklenti tarafından hem ZAP hem de Burp Suite'e kullanılabilecek bayrak. Daha fazla ayrıntı için aşağıdaki bağlantılara bakın.

- OWASP ZAP için ASD Eklenti Evi

- PortSwigger Burp için ASD Eklenti Evi

Tools (Araçlar)

- OWASP Zed Saldırı Proxy (ZAP)
- Burp Suite'in
- Fiddler

Referance (Referanslar)

- RFC 2616 – Hipertext Transfer Protokolü – HTTP 1.1
- OWASP Saldırı Yüzey Dedektörü