**JavaScript Dosyalarındaki Gizli Hazineler: getJS Nedir ve Nasıl Kullanılır?**

Modern bir web sitesine girdiğimizde, gördüğümüz tasarımların, tıkladığımız düğmelerin ve kullandığımız interaktif özelliklerin arkasındaki beyin, büyük ölçüde JavaScript'tir. Bu JS dosyaları, bir uygulamanın iş mantığını, API'larla nasıl iletişim kurduğunu ve verileri nasıl işlediğini barındırır. Geliştiriciler için bu kodlar birer yapı taşıyken, siber güvenlik uzmanları için ise adeta birer hazine haritasıdır. Çünkü bu dosyalarda bazen unutulmuş API anahtarları, gizli yönetici yolları veya henüz yayınlanmamış özelliklere ait ipuçları saklı olabilir. İşte bu hazine haritalarını sistematik olarak toplamak için tasarlanmış aracın adı **getJS**'tir.

Bu yazıda, getJS aracının ne olduğunu, neden her sızma testçisinin ve bug bounty avcısının kullanması gerektiğini, nasıl kurulduğunu ve bu araçla JavaScript dosyalarından nasıl değerli bilgiler çıkarabileceğinizi etkili örneklerle inceleyeceğiz.

**1. getJS Aracı Nedir?**

**getJS**, Go programlama dili ile yazılmış, son derece hızlı ve basit bir komut satırı aracıdır. Temel ve tek bir görevi vardır: kendisine verilen bir alan adı (domain) veya alan adı listesindeki web sayfalarını tarayarak, bu sayfalarda bulunan tüm JavaScript (.js) dosyalarının URL'lerini çıkarmak ve listelemek.

Araç, bir web sitesinin HTML kaynak kodunu analiz eder, <script src="..."> etiketlerini bulur ve bu etiketlerde belirtilen tüm JavaScript dosyalarının tam yollarını size sunar. Bunu inanılmaz bir hızla ve eş zamanlı (concurrent) olarak yaparak, yüzlerce alan adını dakikalar içinde taramanıza olanak tanır.

**2. Neden Kullanılır?**

Sadece JS dosyalarının linklerini toplamak neden bu kadar önemli? Çünkü bu dosyalar, bir uygulamanın görünmeyen yüzü hakkında paha biçilmez bilgiler içerir:

* **Gizli Bilgilerin İfşası (Information Disclosure):** En kritik kullanım amacı budur. Geliştiriciler bazen yanlışlıkla API anahtarlarını, gizli token'ları, Amazon S3 bucket isimlerini veya veritabanı bağlantı bilgilerini doğrudan JS kodunun içine yazabilirler.
* **API Uç Noktalarını Keşfetme:** JS kodları, uygulamanın arka planda iletişim kurduğu API yollarını (/api/v2/users, internal.api.company.com/data) içerir. Bu, belgelenmemiş veya gizli API'ları keşfetmek için harika bir yöntemdir.
* **Yeni Alan Adları ve Alt Alan Adları Bulma:** Bazen JS dosyaları, farklı alt alan adlarına veya henüz keşfedilmemiş başka alan adlarına referanslar içerebilir.
* **Zafiyet Analizi için Kaynak Toplama:** Toplanan JS dosyaları, DOM XSS gibi istemci taraflı zafiyetleri bulmak için manuel olarak veya semgrep gibi statik kod analiz araçlarıyla incelenebilir.
* **Kullanılan Teknolojileri Anlama:** Bir JS dosyasına bakarak, uygulamanın hangi kütüphaneleri (örneğin, eski bir jQuery sürümü) veya framework'leri (React, Vue, Angular) kullandığını anlayabilir ve bu teknolojilere özgü bilinen zafiyetleri arayabilirsiniz.

**3. Nasıl Kurulum Yapılır?**

getJS, Go ile yazıldığı için en kolay kurulum yöntemi yine Go üzerinden yapılır.

**Yöntem 1: Go ile Kurulum (Önerilen ve En Güncel Yöntem)**

Sisteminizde Go kuruluysa, tek bir komutla kurulum yapabilirsiniz.

go install github.com/003random/getJS@latest

Bu komut, getJS'in en son sürümünü indirip derleyecek ve sisteminizde doğrudan getjs komutunu kullanabilmenizi sağlayacaktır.

**Yöntem 2: GitHub "Releases" Üzerinden İndirme**

Eğer Go kurmak istemiyorsanız, projenin [GitHub Releases sayfasından](https://www.google.com/url?sa=E&q=https%3A%2F%2Fgithub.com%2F003random%2FgetJS%2Freleases) işletim sisteminize (Linux, Windows, macOS) uygun, önceden derlenmiş dosyayı indirebilirsiniz. İndirdikten sonra bu dosyayı çalıştırılabilir hale getirip PATH dizinlerinize ekleyerek kullanabilirsiniz.

**4. Kullanım Parametreleri Nedir?**

getJS minimalist bir araçtır ve kullanımı oldukça basittir. İşte anahtar parametreleri:

* -i veya --input: Taranacak alan adlarının veya URL'lerin bulunduğu dosyanın yolunu belirtir.
* -o veya --output: Bulunan JS dosyası URL'lerinin kaydedileceği dosyanın adını belirtir.
* -c veya --concurrency: Eş zamanlı çalışacak thread (iş parçacığı) sayısını belirtir. Varsayılan 20'dir. (Örn: -c 100)
* -v veya --verbose: Tarama sırasında hangi hedefin tarandığı gibi daha ayrıntılı bilgileri gösterir.
* --complete: Çıktıda sadece JS dosyasının URL'sini değil, aynı zamanda o JS dosyasının hangi kaynak URL'de bulunduğunu da gösterir. Bu, bağlamı anlamak için çok faydalıdır.
* --resolve: Sadece ana sayfadan değil, aynı zamanda bulunan diğer JS dosyalarının içinden de yeni JS dosyaları bulmaya çalışır.

**5. Etkili Kullanım İçin 5 Örnek ve Açıklamaları**

getJS'in asıl gücü, diğer komut satırı araçlarıyla zincirleme (chaining) kullanıldığında ortaya çıkar.

**Örnek 1: Tek Bir Hedefi Hızlıca Tarama**

En basit kullanım, standart girdiden (stdin) tek bir hedefi okumaktır.

echo "https://hedefsite.com" | getjs -v

* **Açıklama:** Bu komut, hedefsite.com adresini getjs'e girdi olarak verir. -v parametresi sayesinde, tarama sırasında ilerlemeyi görebilirsiniz. Çıktıda hedefsite.com ana sayfasında bulunan tüm .js dosyalarının URL'leri listelenir.

**Örnek 2: Dosyadaki Alt Alan Adı Listesini Tarama ve Sonuçları Kaydetme**

Bir bug bounty programında, genellikle yüzlerce alt alan adınız olur.

getjs -i subdomains.txt -o js\_urls.txt -c 50

* **Açıklama:**
  + -i subdomains.txt: subdomains.txt dosyasındaki her bir alt alan adını (örneğin api.hedefsite.com, dev.hedefsite.com) tek tek okur.
  + -o js\_urls.txt: Bulduğu tüm JS dosyası URL'lerini js\_urls.txt adlı dosyaya kaydeder.
  + -c 50: Taramayı 50 thread ile yaparak süreci hızlandırır.

**Örnek 3: Keşif Zinciri - Subfinder + Httpx + getJS**

Bu, gerçek bir sızma testinde veya bug bounty avında kullanılan klasik bir iş akışıdır.

subfinder -d hedefsite.com -silent | httpx -silent | getjs -o all\_js\_files.txt

* **Açıklama (Adım Adım):**
  1. subfinder -d hedefsite.com -silent: subfinder aracı, hedefsite.com için tüm olası alt alan adlarını bulur.
  2. |: subfinder'ın çıktısını (alt alan adı listesi) bir sonraki komuta girdi olarak gönderir.
  3. httpx -silent: httpx aracı, bu listedeki alan adlarından hangilerinin "canlı" olduğunu (yani geçerli bir web sunucusu çalıştırdığını) kontrol eder.
  4. |: httpx'in çıktısını (sadece canlı olan sitelerin listesi) getjs'e gönderir.
  5. getjs -o all\_js\_files.txt: Son olarak getjs, sadece canlı olan siteleri tarayarak bulduğu tüm JS dosyalarını tek bir dosyada toplar.

**Örnek 4: Bağlamı Anlamak İçin Detaylı Çıktı Alma**

Bir JS dosyasının hangi sayfada bulunduğunu bilmek, zafiyeti anlamak için kritiktir.

cat live\_targets.txt | getjs --complete

* **Açıklama:** --complete parametresi sayesinde çıktı şu formatta olur: [kaynak\_url] -> js\_dosyasi\_url. Örneğin: [https://hedefsite.com/login.php] -> https://hedefsite.com/assets/login.js. Bu, login.js dosyasının sadece giriş sayfasında kullanıldığını ve muhtemelen kimlik doğrulama mantığı içerdiğini anlamamızı sağlar.

**Örnek 5: Bulunan JS Dosyalarında Otomatik Olarak Sır (Secret) Arama**

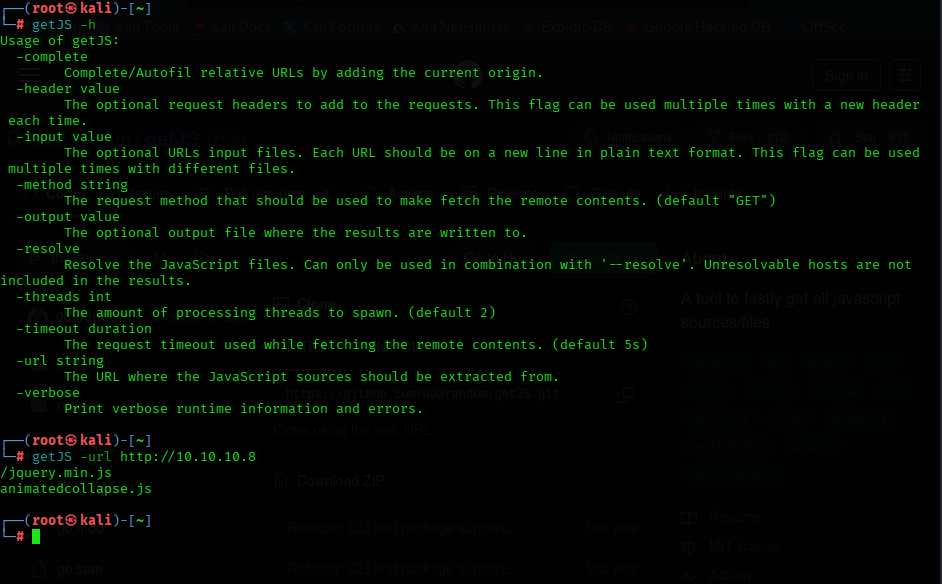
getJS'in asıl amacı olan "hazine avı"nı otomatikleştirelim.

cat all\_js\_files.txt | xargs -n 1 -I {} curl -s -k {} | grep -E "api\_key|secret|S3\_BUCKET|token|password"

* **Açıklama (Adım Adım):**
  1. cat all\_js\_files.txt: Önceki adımlarda topladığımız JS dosyası URL'lerini listeler.
  2. |: Bu listeyi xargs'a gönderir.
  3. xargs -n 1 -I {} curl -s -k {}: xargs, listedeki her bir satırı ({}) alır ve curl komutunu kullanarak o URL'nin içeriğini sessizce (-s) ve SSL hatalarını yoksayarak (-k) indirir.
  4. |: İndirilen her JS dosyasının içeriğini grep'e gönderir.
  5. grep -E "...": grep, gelen kod içinde api\_key, secret, token gibi anahtar kelimelerden herhangi biri geçiyorsa o satırı ekrana basar. Bu komut, binlerce JS dosyasını saniyeler içinde potansiyel sırlar için tarar.

**Örnek: GetJS ile JavaScript Dosyalarının Toplanması**

GetJS aracı, bir web sitesine ait tüm **JavaScript kaynak dosyalarını hızlıca toplamak** için kullanılan bir bilgi toplama (reconnaissance) aracıdır. Bu dosyalar genellikle istemci tarafında çalışan mantığı ve potansiyel olarak hassas verileri içerdiğinden dolayı analiz açısından kritiktir. Aşağıda, http://10.10.10.8 adresine yönelik yapılan bir GetJS taraması yer almaktadır.

  
İlk komutta getJS -h ile yardım menüsü çağrılmış ve aracın tüm parametreleri incelenmiştir. Ardından getJS -url http://10.10.10.8 komutu çalıştırılarak belirtilen hedefteki tüm JavaScript dosyaları listeletilmiştir. Sonuç olarak jquery.min.js ve animatedcollapse.js dosyalarının varlığı tespit edilmiştir. Bu dosyalar, XSS veya DOM tabanlı güvenlik açıklarını analiz etmek için incelenebilir. Ayrıca uygulamanın istemci tarafındaki işleyişine dair ipuçları da bu dosyaların içeriğinde bulunabilir.

**Sonuç**

getJS, basitliğin ne kadar güçlü olabileceğinin mükemmel bir kanıtıdır. Tek bir amaca odaklanmış bu araç, modern web uygulamalarının karmaşık yapısı içinde gizlenmiş değerli bilgileri ortaya çıkarmak için ilk ve en önemli adımlardan birini atmanızı sağlar. Bir uygulamanın saldırı yüzeyini anlamak, gizli API'larını keşfetmek ve geliştiricilerin dalgınlıkla bıraktığı sırları bulmak için getJS vazgeçilmez bir yardımcıdır.

Her zaman olduğu gibi, bu aracı etik kurallar çerçevesinde ve **yalnızca test etme izniniz olan sistemlerde** kullanmayı unutmayın. Doğru kullanıldığında getJS, siber güvenlik cephaneliğinizin en keskin araçlarından biri olacaktır.