Arabellek Taşmaları

Amaç

Aşağıdakileri sağlamak.

- Uygulamalar kendilerini hatalı bileşenlerin etkilerine açık bırakmazlar
- Uygulamalar mümkün olduğunca az arabellek taşmaları barındırırlar
- Arabellek taşmalarına nispeten daha bağışık diller ve çatıların (frameworks) kullanılmasını teşvik etmek

Etkilenen Platformlar

Aşağıdaki dikkate değer istisnalar dışında hemen hemen bütün platformlar:

- J2EE – yerli(native) kodlar veya sistem çağrıları (system calls) çağrılmadıkça
- .NET – (P/Invoke veya COM Interop kullanımı gibi) güvensiz veya yönetilmeyen (unmanaged) kodlar çağırılmadıkça
- PHP – C veya C++ ile yazılmış saldırıya açık PHP uzantıları ve dış programlar çağırılmadıkça

İlgili COBIT Konuları

DS11.9 – Veri işleme bütünlüğü

Açıklama

Saldırganlar arabellek taşmalarını bir web uygulamasının yürütme yığıtını (execution stack) bozmak için kullanırlar. Saldırgan web uygulamasına dikkatlice oluşturduğu girdiyi göndererek, uygulamanın keyfi kod çalıştırmasına neden olabilir. Saldırganlar hayret verici sayıda ürün ve bileşenlerde arabellek taşmalarını teşhis etmeyi başarmışlardır.

Arabellek taşma kusurları, web uygulamasının kendisinde veya sitenin statik ve dinamik görünüşünü sunan web veya uygulama sunucu ürünlerinde bulunabilirler. Yaygın olarak kullanılan sunucu ürünlerinde bulunan arabellek taşmaları, daha fazla kişi tarafından bilinmeye yatkın olup, bu ürünleri kullanan kullanıcılar için kayda değer bir

risk oluştururlar. Web uygulamaları, resim üretmek için kullanılan grafik kütüphaneleri gibi kütüphaneler (library) kullandığında, kendilerini potansiyel arabellek taşmalarına açık bırakırlar. Yaygın olarak kullanılan ürünlere karşı kullanılan arabellek taşmaları hakkındaki yazılar bir çok yerde bulunabilir.

Arabellek taşmaları, özel geliştirilmiş web uygulama kodlarında bulunabilirler ve hatta web uygulamalarının genellikle yetersiz incelemelerden geçirildiği düşünülürse içlerinde arabellek taşmalarını bulundurmaları daha da olasıdır. Özel geliştirilmiş web uygulamalarına karşı uygulanan arabellek taşmaları bazen ilginç sonuçlara yol açar. Bazı durumlarda büyük girdilerin web uygulamasında veya arka uç veritabanında arızaya yol açtığını ortaya çıkardık. Kusurun ciddiyetine ve çeşidine göre web sitesine karşı bir hizmet dışı bırakma saldırısına sebebiyet vermek mümkündür. Çok büyük girdiler, uygulamanın daha sonra sisteme başarılı bir saldırıya neden olacak detaylı hata mesajları vermesine neden olabilir.

Yığıt (Stack) Taşması

Yığıt taşmaları, “arabellek” taşmalarının en iyi anlaşılan ve en yaygın formudur. Yığıt taşmalarının esasları basittir:

- İki arabellek vardır; keyfi saldırı girdisini barındıran kaynak arabellek ve saldırı girdisini içine alamayacak kadar küçük olan hedef arabellek. İkinci arabellek yığıt üzerinde olmalı ve yığıt üzerindeki fonksiyon geri dönüş (return) adresine yakın denebilecek bir yerde olmalıdır.
- Kusurlu kod, ilk arabelleğin, ikinci arabellek için çok büyük olduğunu kontrol etmez. Düşman veriyi, ikinci arabelleği ve fonksiyon dönüş adresini yok edecek şekilde ikinci arabelleğe kopyalar,
- Fonksiyon geri döndüğünde, CPU yığıt çerçevesini (stack frame) açar ve geri dönüş adresini yığıttan alır. Geri dönüş adresi şimdi kirlenmiştir (değiştirilmiştir) ve saldırı kodunu göstermektedir.
- Uygulamada daha önce çağıran tarafa (previous caller) geri dönmek yerine saldırı kodu çalıştırılır.

Saldırıya açık olup olmadığınızı anlama (yolları)

Eğer programınız:

- Arabellek taşmasından müstarip olan bir dille yazılmış veya bu diller ile yazılmış başka bir programa bağlı ise VE
- Kullanıcıdan girdi alıyorsa VE
- Girdiyi arıtmıyorsa VE
- Kanarya değerleri (canary values) kullanmadan yığıt üzerinde ayrılmış değişkenler kullanıyorsa

Uygulama büyük ihtimalle saldırıya açıktır.

Kendinizi koruma (yolları)

- Çalıştırılmayan yığıtlar kullanabilen sistemler üzerinde programınızı kurun ve işletin (AMD ve Intel x86-64 çipleri ile ilgili 64 bitlik işletim sistemleri (XP SP2 - hem 32 hem 64 bitlik-, Windows 2003 SP1 - hem 32 hem 64 bitlik-, AMD ve x86-64 işlemciler üzerinde 32 ve 64 bit modlarında Linux 2.6.8 sonrası,

OpenBSD - w^x özellikli Intel, AMD, Sparc, Alpha ve PowerPC-, noexec_user_stack özelliği çalışır durumda olmayan Solaris 2.6 ve sonrası) ???)

- C ve C++ dışındaki programlama dillerini kullanın
- Çok uzun girdileri önlemek için kullanıcı girdilerinin geçerliliklerini denetleyin ve önceden anlaşılmış şartlara (mesela A-Z, a-z, 0-9, v.b.) uyduklarına emin olmak için değerleri kontrol edin.
- Eğer C ve C++ dillerinde yazılmış işletim sistemi ve yardımcı programlar kullanılıyorsa, en düşük erişim hakkı prensibini (the principle of least privilege) kullandıklarından emin olun, yığıt ve yığın (heap) taşmalarına karşı koruyucu derleyiciler kullanın ve sistemi yamaları ile güncel tutun.

Yığın Taşması

Yığın taşmaları problemlidirler ve "çalıştırılmayan yığıtları" (no executable stacks) ayarlayabilen CPUlar tarafından tam olarak korunmazlar. Yığın, uygulama yürütme süresi (application run time) tarafından lokal olarak tanımlanmış değişkenler için ayrılmış bir bellek alanıdır.

```
function foo(char *bar) {
      char thingy[128];
      …
}
```

`bar` yığıt üzerinden geçirilmişken (passed) `thingy` için yer, yığın üzerinde ayrılmıştır. Taşma yolları yığıt taşmalarındakiler ile tamamen aynı şekildedir.

Saldırıya açık olup olmadığınızı anlama (yolları)

Eğer programınız:

- Yığın arabellek taşmalarından müstarip olan bir dille yazılmış veya bu diller ile yazılmış başka bir programa bağlı ise VE
- Kullanıcıdan girdi alıyorsa VE
- Girdiyi arıtmıyorsa VE
- Kanarya değerleri kullanmadan yığın üzerinde ayrılmış değişkenler kullanıyorsa

Uygulama büyük ihtimalle saldırıya açıktır.

Kendinizi koruma (yolları)

- C ve C++ dışındaki programlama dillerini kullanın
- Çok uzun girdileri önlemek için kullanıcı girdilerinin geçerliliklerini denetleyin ve şartlara (mesela A-Z, a-z, 0-9, v.b.) uyduklarına emin olmak için değerleri kontrol edin.
- Eğer C ve C++ dillerinde yazılmış işletim sistemi ve yardımcı programlar kullanılıyorsa, en düşük erişim hakkı prensibini kullandıklarından emin olun, yığıt ve yığın (heap) taşmalarına karşı koruyucu derleyiciler kullanın ve sistemi yamaları ile güncel tutun.

Format Dizgisi

Format dizgi arabellek taşmaları, kullanıcının aşağıdakine benzer girdileri girmesinden kaynaklanır:

```
%08x.%08x.%08x.%08x.%08x\n
```

Üstteki saldırı dizgisi yığıt üzerindeki beş kaydı basacaktır. Format dizgileri son derece özelleşmiş arabellek taşmalarıdır ve buna rağmen, uzaktan tamamen ele geçirme dahil aynı saldırıları uygulamak için kullanılabilirler.

Saldırıya açık olup olmadığınızı anlama (yolları)

Eğer programınız:

- Arabellek taşmalarından müstarip olan bir dille yazılmış veya bu diller ile yazılmış başka bir programa bağlı ise VE
- Kullanıcıdan girdi alıyorsa VE
- Girdiyi arıtmıyorsa VE
- printf(), snprintf() ve bunlara benzer veya eş değer diğer fonksiyonları, veya bunları kullanan syslog gibi system servislerini kullanıyorsa

Uygulama büyük ihtimalle saldırıya açıktır.

Kendinizi koruma (yolları)

- C ve C++ dışındaki programlama dillerini kullanın
- Çıktı formatlarını değiştiren, kullanıcı girdilerine izin veren printf() veya benzer fonksiyonları kullanmaktan kaçının
- Format dizgi değişim (meta) karakterlerinin girdilerde kullanılmasını önlemek için kullanıcı girdilerinin geçerliliğini denetleyin
- Eğer C ve C++ dillerinde yazılmış işletim sistemi ve yardımcı programlar kullanılıyorsa, en düşük erişim hakkı prensibini kullandıklarından emin olun, yığıt ve yığın (heap) taşmalarına karşı koruyucu derleyiciler kullanın (tam bir korunma yöntemi değildir) ve sistemi yamaları ile güncel tutun.

Unicode (Evrensel Kod) Taşmaları

Evrensel kod istismarlarının uygulanmaları Anley'in 2002 yılındaki makalesinde gösterildiği gibi, tipik arabellek taşmalarına gore biraz daha zordur, ama evrensel kod kullanarak arabellek taşmalarından korunduğunuzu varsaymak yanlıştır. Evrensel kod taşmalarına örneklerden bir tanesi yıkıcı bir Truva atı olan Code Red'dir.

Saldırıya açık olup olmadığınızı anlama (yolları)

Eğer programınız:

- Arabellek taşmalarından müstarip olan bir dille yazılmış veya bu diller ile yazılmış başka bir programa bağlı ise VE
- Kullanıcıdan Unicode girdisi alıyorsa VE
- Girdiyi arıtmıyorsa VE
- Kanarya değerleri kullanmadan yığıt ve yığın üzerinde ayrılmış değişkenler kullanıyorsa

Uygulama büyük ihtimalle saldırıya açıktır.

Kendinizi koruma (yolları)

- Web, uygulama sunucu ve Internet altyapınızdaki diğer ürünlerinize ait en son hata raporlarını takip edin. Bu ürünlere ait en son yamaları uygulayın.
- Sunucu ürünlerinizde ve özel yapım web uygulamalarınızda arabellek taşma hatalarını bulacak yaygın olarak kullanılan tarayıcılar ile web sitenizi periyodik olarak tarayın.
- Kodunuzu Unicode istismarlarına karşı inceleyin.

Özel yapım (custom) uygulama kodunuz için, güvenilmeyen her kaynaktan gelen girdileri kabul eden bütün kodları incelemeniz ve kodun bu şekildeki bütün girdiler üzerinde büyüklük kontrollerini uyguladığından emin olmanız gerekir.

Bu kontroller, aslında bu tür saldırılara karşı etkilenmeyen yerlerde de yapılmalıdır çünkü yakalanmayan çok uzun girdiler, hizmet dışı bırakma ve diğer operasyonel problemlere de neden olabilirler.

Tamsayı Taşması

Bir uygulama, iki belirlenmiş büyüklükte sayı aldığında ve bunlar ile bir işlem yaptığında, sonuç aynı büyüklükte çıkmayabilir. Mesela, iki 8 bitlik numara olan 192 ve 208 birbirine eklendiğinde ve diğer bir 8 bitlik bir baytta saklanırsa, işlem sonrası meydana çıkan sayı 8 bitlik sonuca sığmayacaktır:

```
 %    1100 0000
+ %   1101 0000
= % 0001    1001 0000
```

En üstteki yarım sözcük (4 bit) atılacaktır ve geriye kalan geçerli bir sonuç değildir. Bu durum bütün diller için bir problem olabilir. Mesela, bir çok hexadecimal çeviriler (conversion) %M0'ı 192'ye "başarılı bir şekilde" çevirecektir. Endişelenilmesi gereken diğer alanlar, dizilim indisleri (array indices) ve örtülü (implicit) kısa matematik işlemleridir.

Saldırıya açık olup olmadığınızı anlama (yolları)

- Özellikle bayt ve short olan işaretli (signed) tamsayıları inceleyin
- Bu değerlerin, + - * / veya % gibi aritmetik işlemler uygulandıktan sonra dizilim indisleri olarak kullanıldığı yerler var mı?
- Kod, negative veya sıfır indisleri ile başa çıkıyor mu?

Kendinizi koruma (yolları)

- .NET: David LeBlanc'ın SafeInt<> C++ sınıfını veya diğer yapılarını kullanın
- .Derleyiciniz destekliyorsa ve tam aksi söylenmemişse, tamsayılar için varsayılı değeri işaretsiz (unsigned) olarak değiştirin. Ne zaman kullanmanız gerekiyorsa işaretsiz (unsigned) tamsayıları kullanın.
- .Eğer diliniz ve çatınız (framework) destekliyorsa değer kümesi kontrolleri kullanın
- .Küçük değerlerle uğraşırken aritmetik işlemlerde dikkatli olun, özellikle taşma, işaret değişiklikleri veya diğer hatalar oluşabilecekken.

Ek okumalar

- Team Teso, *Exploiting Format String Vulnerabilities*
  http://www.cs.ucsb.edu/~jzhou/security/formats-teso.html
- Woo woo and Matt Conover, *Preliminary Heap Overflow Tutorial*
  http://www.w00w00.org/files/articles/heaptut.txt
- Chris Anley, *Creating Arbitrary Shellcode In Unicode Expanded Strings*
  http://www.ngssoftware.com/papers/unicodebo.pdf
- David Leblanc, *Integer Handling with the C++ SafeInt Class*
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure01142004.asp
- Aleph One, *Smashing the Stack for fun and profit*
  http://www.phrack.org/phrack/49/P49-14
- Mark Donaldson, *Inside the buffer Overflow Attack: Mechanism, method, & prevention*
  http://rr.sans.org/code/inside_buffer.php
- *NX Bit*, Wikipedia article
  http://en.wikipedia.org/wiki/NX_bit
- Horizon, *How to bypass Solaris no execute stack protection*
  http://www.secinf.net/unix_security/How_to_bypass_Solaris_nonexecutable_stack_protection_.html
- Alexander Anisimov, *Defeating Microsoft Windows XP SP2 Heap protection and DEP bypass*, Positive Technologies
  http://www.maxpatrol.com/defeating-xpsp2-heap-protection.htm

Dökümanın Aslı : OWASP GUIDE 2.0.1

Çeviri **:** Bedirhan Urgun - bedirhan@webguvenligi.org