SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SİSTEM PROGRAMLAMA 2. PROJE ÖDEVİ

Teslim Tarihi: 30.4.2017 Pazar Saat: 23:59
Teslim yeri: sisprog54@gmail.com
Teslim şekli: Metin dosyası olarak **sisPro2grpXogrY.txt** şeklinde emaile ek olarak gönderilecek. X yerine grup nosu, Y yerine öğretim nosu gelecek.

1. Giriş

Bu proje çalışması, farklı güvenlik açıklarına sahip bir program üzerinde toplam üç saldırı çalışmasını kapsıyor. Edineceğiniz sonuçlar şunları içeriyor:

- Güvenlik açısından zayıf programların kendilerini bellek taşmalarına karşı koruyamadıklarında (buffer overflow), saldırganların güvenlik açıklarını kullanabileceği farklı yolları öğreneceksiniz.
- Bu sayede, programların daha güvenli hale getirilmesinin yanı sıra, programları daha güvenilir hale getirmek için derleyiciler ve işletim sistemleri tarafından sağlanan bazı özelliklerin daha iyi anlaşılması sağlanacak.
- X86-64 makine kodunun yığın ve parametre geçirme mekanizmalarını daha iyi anlayacaksınız.
- X86-64 talimatlarının nasıl kodlandığına dair bilgi birikimi kazanacaksınız.
- `gdb` ve `objdump` gibi hata ayıklama araçlarıyla deneyim kazanacaksınız.

Not: Bu projede, işletim sistemlerinde ve ağ sunucularında güvenlik zayıflıklarını kullanmak için var olan yöntemlerle ilgili ilk deneyime sahip olacaksınız. Amacımız, programların çalışması hakkında bilgi edinmenize yardımcı olmaktır. Bu güvenlik zayıflıklarının doğasını anlayın ve böylece sistem kodu yazarken bunları önleyin. Herhangi bir sistem kaynağına yetkisiz erişim sağlamak için burada gösterilen saldırı biçimlerini kullanmanız asla istenmez.

2. Özellikler

Proje grup olarak yapılacaktır, sonuçlar sisprog54@sakarya.edu.tr adresine grup adına bir kişi tarafından gönderilecektir.

2.1 Dosyaların elde edilmesi

"target.tar" dosyasını SABIS üzerinden indirebilirsiniz. Bu sıkıştırılmış dosyayı açtığınızda karşınıza aşağıdaki dosyalar çıkacak.

- README.txt: Dizinin içeriğini açıklayan bir dosya

- ctarget: *code injection* saldırılana açık bir icra edilebilir program

- cookie.txt: Saldırılarınızda tanımlayıcı olarak kullanacağınız 8 dijitli bir hexodesimal rakam.

- hex2raw: Saldırı stringi üretmek için kullanılacak program.

2.2 Önemli noktalar

Projenizde sadece touch1, touch2 ve touch3 adlı fonksiyonların adresleri ile ilgilenilecektir.

3. Hedef programlar

`ctarget` programı standart girişten (stdin) bir string'i aşağıda verilen getbuf() fonksiyonu ile yapmaktadır.

```
1 unsigned getbuf()
2 {
3       char buf[BUFFER_SIZE];
4       Gets(buf);
5       return 1;
6 }
```

`Gets()` fonksiyonu, standart kütüphane çağrısı `gets()` ile benzerdir - standart girdiden bir string okur ('\n' veya dosya sonu ile sonlandırılmış) ve belirtilen varış yerine (null sonlandırıcıyla birlikte) yani buffer'a kopyalar. Bu kodda, hedefin, BUFFER_SIZE bayt olduğu bildirilen bir karakter buffer'ı (`buf`) olduğunu görebilirsiniz.

`Gets()` ve `gets()` fonksiyonlarının, okunan string'in boyutunun hedef buffer'a depolayacak kadar büyük olup olmadığını tespit etme özelliği yoktur. Yani, ayrılmış belleğin sınırlarını aşarak büyük olasılıkla girdiyi buffer'a kopyalarlar. Kullanıcı tarafından yazılmış ve `getbuf()` fonksiyonu tarafından okunan string yeterince kısa ise, aşağıdaki yürütme örnekleri tarafından gösterildiği gibi `getbuf()` 1 döndürür:

```
unix> ./ctarget
Cookie: 0x1a7dd803
Type string: Keep it short!
No exploit. Getbuf returned 0x1
Normal return
```

Eğer BUFFER_SIZE'dan uzun bir string girilirse aşağıdaki gibi bir hata mesajı alınır:

```
unix> ./ctarget
Cookie: 0x1a7dd803
Type string: This is not a very interesting string, but it has the
property ...
Ouch!: You caused a segmentation fault!
Better luck next time
```

`ctarget` programı aşağıdaki parametreleri alabilir:

-h: Olası argüman listesini yazdırır.

-q: Sonuçları notlama sunucusuna göndermek içindir.

-i FILE: Giriş parametrelerini `stdin` yerine bir dosyadan al.

Aşama	Program	Seviye	Metot	Foksiyon
1	`ctarget`	1	Code Injection	touch1
2	`ctarget`	2	Code Injection	touch2
3	`ctarget`	3	Code Injection	touch3

Exploit string'leriniz genellikle ASCII değerlerine karşılık gelmeyen bayt değerlerini içerecektir. `hex2raw` programı bu ham dizeleri üretmenizi sağlayacaktır.

4. Kod enjeksiyon saldırısı

4.1 Seviye 1

Aşama 1 için yeni kod enjekte etmeyip sadece geri dönüş adresini uyarlamanız gerekmektedir. Aşağıda gösterildiği gibi `getbuf`() fonksiyonu `ctarget` içinden çağrılmaktadır:

```
1 void test()
2 {
3    int val;
4    val = getbuf();
5    printf("No exploit. Getbuf returned 0x%x\n", val);
6 }
```

4. satırdaki `getbuf`() fonksiyonu kendi içindeki `return` ifadesini çalıştırdığında, program normalde test fonksiyonunun 5. Satırından itibaren yürütmeyi sürdürür. Bu davranışı değiştirmek istiyoruz. `ctarget`'in içinde aşağıdaki şekilde C dilinde gösterilen `touch1`() fonksiyonu bulunmaktadır:

```
1 void touch1()
2 {
3    vlevel = 1; /* Part of validation protocol */
4    printf("Touch1!: You called touch1()\n");
5    validate(1);
6    exit(0);
7 }
```

Göreviniz, `getbuf`() fonksiyonu `return` ifadesini icra ederken `test`() fonksiyonuna geri dönmek yerine `ctarget` içindeki `touch1`() fonksiyonununa gitmesini sağlamaktır. Exploit string'inin doğrudan bu aşamayla ilgili olmayan yığın bölümlerini bozabileceğini unutmayın, ancak yapacağınız değişiklik `touch1()` fonksiyonunun çalışmasını sağlayacağından herhangi bir sorun olmayacaktır.

Yol gösterme:

- Exploit stringini bu düzey için tasarlarken gerekli tüm bilgiler, `ctarget'in` assembler hali incelenerek belirlenebilir. Bu dis-assembled sürümü elde etmek için objdump -d kullanın.
- Buradaki fikir, `touch1`() fonksiyonunun başlangıç adresinin konumlandırılmasıdır, böylece `getbuf`() içindeki kodun sonundaki ret komutu, kontrolü `touch1()`'e döndürecektir.
- Byte sıralamaya dikkate etmelisiniz (little endian, big endian…).
- `gdb`'yi kullanarak, doğru şeyi yaptığınızdan emin olmak için `getbuf()`'ın son birkaç komutunu adım adım icra ettirebilirsiniz.
- `Getbuf`() için yığın çerçevesindeki buf'ın yerleşim bilgisi ile BUFFER_SIZE değerini tespit etmelisiniz. Bunun için dis-assembled kodu incelemeniz gerekecek.

4.2 Seviye 2

Bu aşamada, exploit stringi içine az miktarda kod enjekte etmeniz gerekiyor. `ctarget` içinde aşağıdaki C gösterimine sahip `touch2`() fonksiyonu bulunmaktadır:

```
1 void touch2(unsigned val)
2 {
3    vlevel = 2; /* Part of validation protocol */
4    if (val == cookie) {
5         printf("Touch2!: You called touch2(0x%.8x)\n", val);
6         validate(2);
```

```
7     } else {
8         printf("Misfire: You called touch2(0x%.8x)\n", val);
9         fail(2);
10    }
11    exit(0);
12 }
```

Göreviniz `ctarget`'ın test fonksiyonuna geri dönmesi yerine `touch2`() koduna gitmesini sağlamaktır. Ancak, burada geçerli bir cookie'yi argüman olarak göndermeniz de gerekiyor.

Yol gösterme:

- Enjekte edilen kodun adresinin bayt stringi'ni + ret adresini kontrolü döndürmek istediğiniz yere aktaracak şekilde konumlandırmalısınız.
- Intel mimarisinde ilk parametre `%rdi` saklayıcısı ile gönderiliyor.
- Enjekte edilen kodunuz, bu saklayıcıya geçerli cookie'yi yüklemelidir ve kontrolü `touch2()`'deki ilk komuta aktarmak için bir ret komutu kullanmalıdır.
- Exploit kodunuzda `jmp` veya `call` komutlarını kullanmaya kalkışmayın çünkü bu komutlarla hedef adresleri hesaplayarak kodlamak zordur. Tüm kontrol aktarımları için `ret` komutlarını kullanın.

4.3 Seviye 3

Bu aşamada yine kod enjeksiyonu saldırısı yapacaksınız, ancak bu sefer bir string'in parametre olarak gönderilmesi düşünülmüştür. `ctarget` dosyasında, aşağıdaki C gösterimlerine sahip olan `hexmatch()` ve `touch3`() fonksiyonları için kod bulunmaktadır:

```
1 /* Compare string to hex represention of unsigned value */
2 int hexmatch(unsigned val, char *sval)
3 {
4    char cbuf[110];
5    /* Make position of check string unpredictable */
6    char *s = cbuf + random() % 100;
7    sprintf(s, "%.8x", val);
8    return strncmp(sval, s, 9) == 0;
9 }
10
11 void touch3(char *sval)
12 {
13    vlevel = 3; /* Part of validation protocol */
14    if (hexmatch(cookie, sval)) {
15        printf("Touch3!: You called touch3(\"%s\")\n", sval);
16        validate(3);
17    } else {
18         printf("Misfire: You called touch3(\"%s\")\n", sval);
19        fail(3);
20    }
21    exit(0);
22 }
```

Göreviniz `ctarget`'in teste dönmek yerine `touch3()` kodunu çalıştırmasını sağlamaktır. `touch3` fonksiyonuna, cookie'nin parametre olarak gönderildiği string'i sanki normal bir giriş parametresi gibi oluşturmalısınız.

Yol gösterme:

- Exploit string'inize cookie'nin bir string gösterimini eklemeniz gerekecek. String, önde gelen "0x" olmadan sekiz hexodesimal dijitten (en yüksekten en az anlamlıya doğru sıralanmalı) oluşmalıdır.
- C dilinde string bir bayt dizisi ve ardından 0 değeri olan bir dizidir. İhtiyacınız olan karakterlerin bayt karşılıklarını görmek için herhangi bir Linux makinesinde "man ascii" yazınız.
- Enjekte edilecek kod `%rdi` saklayıcısına parametre olarak gönderilecek string'in adresini yüklemelidir.
- `hexmatch` ve `strncmp` fonksiyonları çağırıldığında, `getbuf`() tarafından kullanılan buffer aracılığı ile yığın üzerine veri yazarlar. Sonuç olarak, cookie'nin string gösterimini nereye yerleştirdiğinize dikkat etmeniz gerekiyor.

Geri döndürülecekler:

Bir metin dosyasında her bir seviye için parametre olarak gönderilen string bir dosyaya konup geri döndürülecektir. Dosyanın adı **sisPro2grpXogrY.txt** şeklinde olmalıdır.

Örnek:

SEVİYE 1:

Parametre: 41 41 41 ….. 41

SEVİYE 2:

Parametre: 41 41 …. 41

SEVİYE 3:

Parametre: c7 48 1e d6……41