Fuzzzzing

written by Mert SARICA | 14 January 2010

Bu seferki yazımı yazmak için biraz geç kaldım, geç olsun ama güç olmasın diyerek hemen konuya giriyorum.
Güvenlik e-posta listelerine üye iseniz her gün onlarca yeni güvenlik zaafiyetlerinin keşfedildiğini görebilirsiniz. Bunların nasıl keşfedildiğini biraz araştıracak olursak genellikle kaynak kodu açık olmayan uygulamalardaki güvenlik zaafiyetlerinin çoğunun fuzzing ile keşfedildiğini görebiliriz. Fuzzingi baştan sona anlatmaya kalkacak olursam kitap yazmam gerekir o nedenle kısaca özetleyip örnek bir fuzzer ve güvenlik açığının keşfedilmesi ile ilgili bir video ile yazımı tamamlayacağım.

Fuzzing'e kabaca hedef uygulamada hataya sebebiyet vermek amacıyla üretilen ve hedefe gönderilen veri diyebiliriz. Fuzzerlar generation ve mutation olarak ikiye ayrılmaktadır. Generation fuzzerları elimizde örnek bir veri olmadan oluşturduğumuz ve hedefe gönderdiğimiz, mutation fuzzerları ile generation fuzzerların aksine örnek bir veriden (örnek bir xls dosyası veya bmp dosyası) türetilen ve hedefe gönderilen veriler olarak düşünebiliriz. Generation fuzzerlara örnek vermek gerekirse bir http sunucusu düşünelim ve port 80'den sunucuya bağlanarak farklı boyutlarda ve karakter setinden rastgele oluşturulan ve hedefe gönderilen veriler olarak düşünebiliriz. Tabii hedef http sunucusu GET/POST vb. benzer komutlar beklediği için parserından geçmeyerek paketler direk çöpe gidecektir.
Mutation fuzzerlar ise rastgele oluşturulmuş bir paket yerine capture edilmiş bir paketten oluştuğu için (örneğin GET /AAAAA… HTTP/1.1) hedef sistem üzerinde soruna yol açma ihtimali generation fuzzerlara göre daha yüksektir. Fuzzerlar ile keşfedilebilecek güvenlik zaafiyetlerinin başında buffer overflow, integer overflow, format string zaafiyetleri gelmektedir.
Fuzzing yapabilmek için haliylen ya veri üreten ve gönderen kendi fuzzerınızı yazacaksınız ya da hali hazırda yazılmış fuzzerlardan faydalanacaksınız. Ufak bir araştırma yaptığınızda hedef programa uygun fuzzerlar bulmak mümkün, örneğin dosya formatını işleyen programları test edecekseniz filefuzz, ağ uygulaması test edecekseniz smudge, framework üzerinde kendi fuzzerınızı geliştirecekseniz sulley, peach vb. programlardan faydalanabilirsiniz.

Sadede gelecek olursam, geçtiğimiz Pazar sabahı bloga ne yazsam ne yazsam diye hindi gibi düşünürken bir fuzzer kodlasam bir de bu fuzzer ile

keşfedilmiş 0day güvenlik zaafiyeti yayınlasam tadından yenmez dedim ve download.com sitesinde hedef program aramaya koyuldum. Sitede biraz gezdikten sonra mediaplayer kategorisinde yer alan, CNET editörleri tarafından 5 yıldız almış ve 1,275,469 defa indirilmiş BS.Player uygulamasına göz atmaya karar verdim. Programı yükledikten sonra bu program ile ilişkili dosya uzantılarını bulmam gerekiyordu bu nedenle hemen Windows XP'de Windows Explorer'da, Tools -> Folder Options -> File Types listesinde yer alan uzantılara baktım ve .bsi uzantısı ilk gözüme çarpanı oldu. Program ile gelen örnek BSI uzantılı bir dosya olmadığı için google üzerinde yaptığım araştırmada bu sayfadaki yazı dikkatimi çekti. Mutation fuzzer için örnek BSI dosyasını bulmuştum ve sıra fuzzer yazmaya gelmişti.

Oturup şip şak python ile kod yazmaya alışmış biri olarak hemen oturdum ama bu sefer hemen kalkamadım. Akşam saatlerine kadar hem kod yazdım hemde programda hata ortaya çıktığında monitör edebilmek için araştırmalar yaptım ve sonunda fuzzing yapabilmek için aşağıdaki programı hazırladım. Monitor edebilmek içinse Sehloger adında bir program buldum.

```
import os
import re
import time
import sys

if sys.platform == 'linux' or sys.platform == 'linux2':
        clearing = 'clear'
else:
        clearing = 'cls'

os.system(clearing)

print "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-="
print "| Simple Fuzzer | Mert SARICA | http://www.mertsarica.com |"
print "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-="

target_process = ' "C:\\Program Files\\Webteh\\BSplayer\\bsplayer.exe"'
# target_process = ' "bsplayer.exe"'
target_file = "test.bsi"
seh_handler = "Sehloger.exe"
sleeptime = 1
```

```
logger = "SEHLog.txt"

debugger = seh_handler + target_process

variables = ["Version", "Title", "FName", "Sub1", "Font", "SubPos",
"FullScreen", "Skin", "Lang", "Aspect", "RunHD", "ExitAtEnd"]

for i in range(0, len(variables)-1):

    re1= "(" + variables[i] + ")"
    re2='(=)'   # Any Single Character 1
    re3='((?:S+))'      # Rest

    readfile = open(target_file, "r")
    tempfile = target_file.split(".")
    tempfile = tempfile[0] + str(i) + "." + tempfile[1]

    print "\nTesting:", variables[i] + "\n"

    txt = readfile.read()
    readfile.close()

    rg = re.compile(re1+re2+re3,re.IGNORECASE|re.DOTALL)
    m = rg.search(txt)

    if m:
        word1=m.group(1)
        c1=m.group(2)
        word2=m.group(3)
        entry = word1+c1+word2

        for m in range(1,10):
            os.system(debugger)
            time.sleep(sleeptime)
            writefile = open(tempfile, "w")
            bof = word1+c1+(("A"*128)*m)

            text = txt.replace(entry, bof)
            writefile.write(text)
```

```
            writefile.close()

            bof_check = target_process + " " + tempfile
            os.system(bof_check)
            time.sleep(sleeptime)

            logfile = open(logger, "r")
            log = logfile.read()

            if log.find("41414141") > 0:
                print "\nGray area detected, responsible disclosure or dark
side padawan? :)\a\n"
                print "Suspucious file:", (tempfile)
                sys.exit()

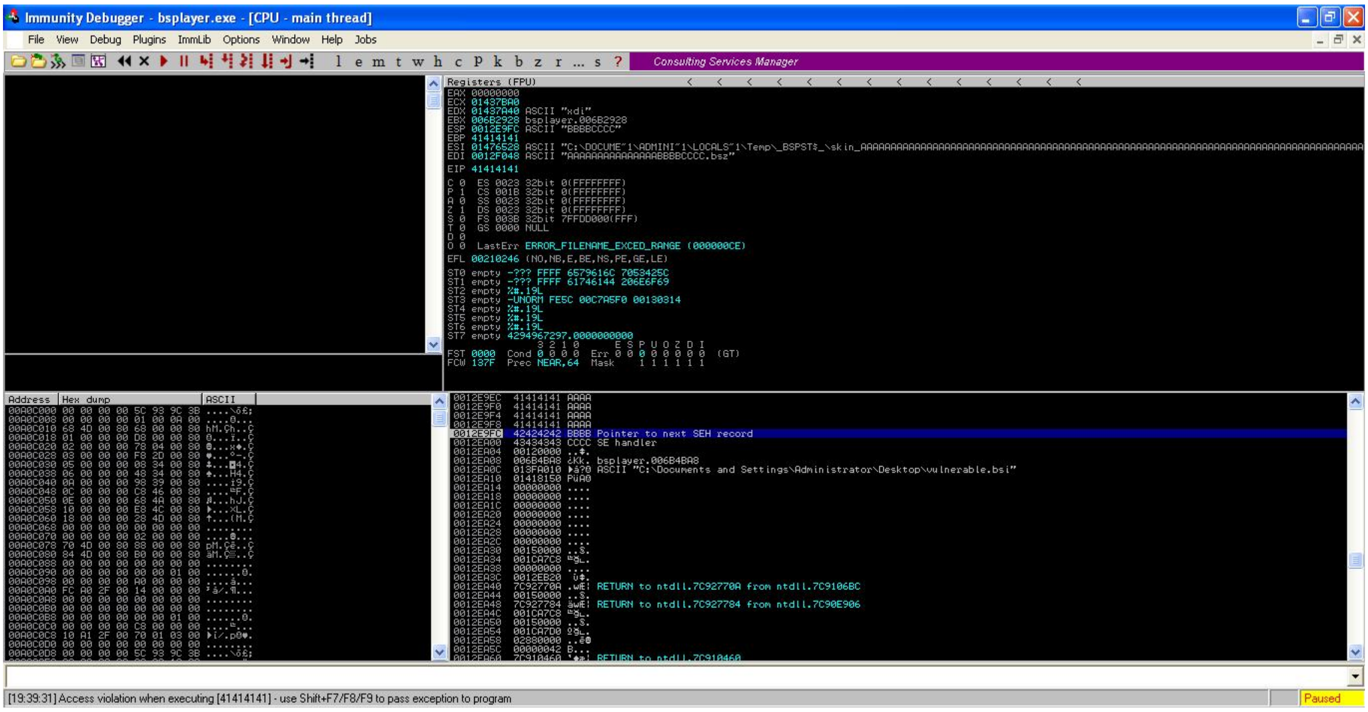
            os.system("taskkill.exe /IM bsplayer.exe")
            time.sleep(sleeptime)
```

Yukarıdaki python kodu örnek BSI dosyası içerisinde yer alan tanımlamaları alıyor ve her değer yerine 128 ve katı sayıda A koyuyor, dosyayı kayıt ediyor ve programı exception handler olarak tasarlanmış olan Sehloger programı ile çalıştırıyor. Sehloger programı ise programda herhangi bir hata olması durumunda o anki register değerlerini log olarak yazıyor, daha sonra yazdığım program ise bu log dosyası içerisinde 0x41414141 bulur ise mutlu sona ulaştığını haber veriyor.

Böyle kuru kuru anlatmakla yetinmeyerek, fuzzerın nasıl çalıştığını ve fuzzer ile nasıl 0 day SEH overwrite güvenlik zaafiyeti keşfettiğimi içeren ufak bir video hazırladım.

Fuzzer programına buradan, SEH overwrite güvenlik zaafiyetini tetikleyen koda ise buradan ulaşabilirsiniz.

SEH overwrite güvenlik zaafiyetine ait ekran görüntüsü ise aşağıdadır.


Immunity Debugger - bsplayer.exe - [CPU - main thread]
File View Debug Plugins ImmLib Options Window Help Jobs
Consulting Services Manager
Registers (FPU)
EIP 41414141
LastErr ERROR_FILENAME_EXCED_RANGE (000000CE)
0012E9FC 42424242 BBBB Pointer to next SEH record
0012EA00 43434343 CCCC SE handler
[19:39:31] Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program
Paused