HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Ispitivanje sigurnosti klijentskih aplikacija

CCERT-PUBDOC-2005-10-138

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.
Rezultat toga rada ovaj je dokument koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, **www.lss.hr**- laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Ovaj dokument predstavlja vlasništvo CARNet-a (CARNet CERT-a). Namijenjen je za javnu objavu, njime se može svatko koristiti, na njega se pozivati, ali samo u originalnom obliku, bez ikakvih izmjena, uz obavezno navođenje izvora podataka. Korištenje ovog dokumenta protivno gornjim navodima, povreda je autorskih prava CARNet-a, sukladno Zakonu o autorskim pravima. Počinitelj takve aktivnosti podliježe kaznenoj odgovornosti koja je regulirana Kaznenim zakonom RH.

Sadržaj

1. Uvod

Fuzzing je dobro poznata i provjerena tehnika pronalaženja ranjivosti, odnosno sigurnosnih propusta u različitim programskim rješenjima. *Fuzzing* tehnika aplikaciju testira tako da na jedan ili više podatkovnih ulaza šalje različite unose koji bi mogli uzrokovati nepredvidivo ponašanje programa, odnosno ukazati na potencijalni sigurnosni nedostatak. *Fuzzing* tehnika datira još iz devedesetih godina prošlog stoljeća kada je tim sigurnosnih stručnjaka analizirao integritet programa na Unix operacijskim sustavima. Tijekom grmljavine jedan od članova tima pokušao je koristiti određene Unix alate preko modemske veze. Zbog šuma na liniji, programi su umjesto legitimnih podataka generirali pseudo-slučajne nizove koji su uzrokovali neobjašnjivo "rušenje" velikog broja različitih programa. Na temelju ovog otkrića razvijen je program pod nazivom `fuzz`, koji je provjeravao sigurnost Unix aplikacija pomoću pseudo-slučajnih podataka. Ovom metodom otkriven je velik broj ranjivosti kao što su preljevi spremnika, preljevi cjelobrojne vrijednosti, *format string* propusti, i sl., koji su tek desetak godina kasnije prepoznati kao ozbiljna prijetnja računalnoj sigurnosti.

Zbog relativne jednostavnosti i brzine, *fuzzing* tehnika vrlo je brzo postala iznimno popularna tehnika za otkrivanje sigurnosnih propusta. Proces *fuzzing* testa na aplikaciji može u vrlo kratkom vremenu otkriti kritične ranjivosti za koje bi pregledavanje izvornog koda ili analiza binarnih datoteka zahtijevalo mnogo više utrošenog vremena i resursa. Tako je danas na Internetu moguće pronaći nekoliko besplatnih i komercijalnih *fuzzing* alata. Najpopularniji besplatni alati su `SPIKE` (http://www.immunitysec.com), mrežni *fuzzer* kojeg je napisao poznati istraživač Dave Aitel te alat pod nazivom `SMUDGE` (http://www.felinemenace.org) napisan od strane stručnjaka za računalnu sigurnost pod nadimkom Ned. Od komercijalnih riješenja popularan je program pod nazivom `CHAM` (engl. ***C**ommon **H**acking **A**ttack **M**ethods*), *fuzzer* koji se distribuira u sklopu `Retina` alata za provjeru ranjivosti razvijenog od srane renomirane tvrtke `Eeye` (http://www.eeye.com). Treba napomenuti da postoji i određeni broj anonimnih *fuzzer* programa koje se razvili pojedinci koji se bave istraživanjem na području računalne sigurnosti, međutim ovi alati nisu toliko poznati u javnosti.

Iznimnu popularnost u posljednje je vrijeme stekla tehnika *fuzzinga* klijentskih aplikacija kao što su Web preglednici, media *playeri*, programi za obradu teksta, i sl. *Fuzzing* klijentskih aplikacija uzeo je maha kada je prije nekoliko godina poznati sigurnosni stručnjak Michal Zalewski objavio iznimno jednostavan *fuzzer* za provjeru sigurnosti Web preglednika pod nazivom `Mangleme`. Iako jednostavan, taj *fuzzer* je otkrio brojne preljeve spremnika i slične ranjivosti u gotovo svim popularnijim Web preglednicima (npr. `Internet Explorer`, `Mozilla`, `Lynx`, `Opera`, i sl.). Također, nedavno je tvrtka `iDefense` objavila `FileFuzz` program namijenjen *fuzzingu* datoteka, odnosno programa koji te datoteke otvaraju i obrađuju. Pomoću tog alata moguće je otkriti ranjivosti u aplikacijama kao što su `RealPlayer`, `MS Word`, i sl. Prije nekoliko mjeseci prije spomenuti istraživač Michal Zalewski objavio je također *fuzzer* za testiranje programskog koda za prikaz JPEG slika unutar Web preglednika. Iako je *fuzzer* opet vrlo jednostavan, otkriveno je nekoliko kritičnih ranjivosti unutar `Internet Explorer` Web preglednika, u dijelu koda za obradu spomenutih JPEG slika.

Iz prije navedenih činjenica i primjera, vidljivo je da je popularnost programa ovog tipa u iznimnom porastu te da učinkovitost i efikasnost *fuzzer* programa ovisi više o idejama i domišljatosti programera nego o njegovoj kompleksnosti. U dokumentu je prikazano korištenje i izrada *fuzzer* alata za ispitivanje sigurnosti klijentskih aplikacija, te neki konkretni primjeri otkrivanja ranjivosti koje su već objavljene.

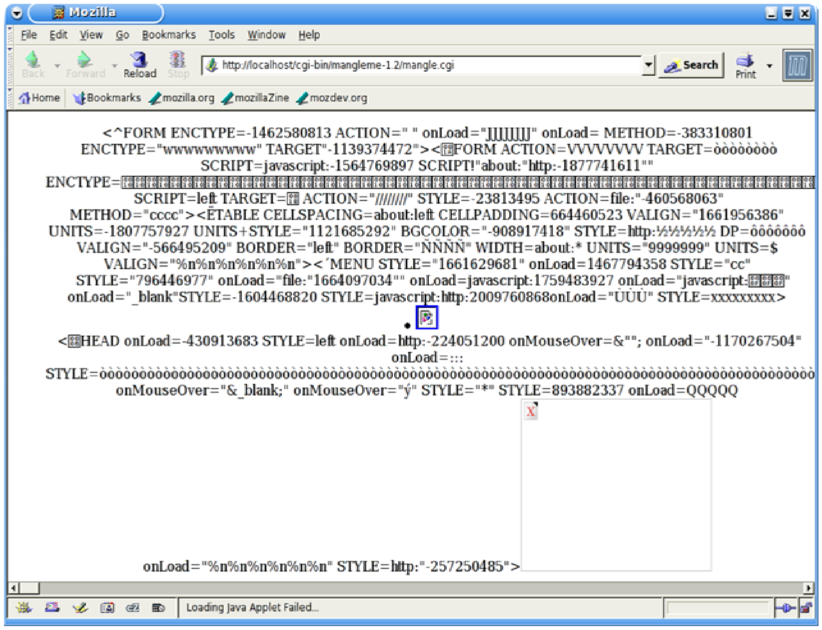
2. Fuzzing Web preglednika

Popularizacijom World Wide Weba, Web preglednici (engl. *browser*) su postali neizbježna i jedna od najvažnijih klijentskih aplikacija za pristup Internetu. Iz tog razloga sigurnost Web preglednika vrlo je važan segment iz perspektive sigurnosti za krajnjeg korisnika. Nažalost, trenutna situacija u području sigurnosti Web preglednika prilično je nezahvalna. Osim `Internet Explorer`-a, ranjivosti se sve češće otkrivaju i u alternativnim Web preglednicima kao što su `Mozilla`, `Opera` i sl., što potvrđuje da niti njihovo korištenje korisnike ne štiti od malicioznih aktivnosti koje prijete s Interneta.

Za testiranje i provjeru sigurnosti Web preglednika koristiti će se ranije spomenuti `Mangleme` alat razvijen od strane Michala Zalewskog, a koji se može pronaći na adresi http://lcamtuf.coredump.cx/soft/mangleme.tgz. Alat je izveden kao CGI (engl. ***C**ommon **G**ateway **I**nterface*) skripta koja se postavlja na Web poslužitelj, pri čemu Web pregledniku šalje različite varijacije neobičnih i neočekivanih HTML oznaka. Nakon inicijalnog testiranja koje je proveo sam autor alata, otkrivene su ranjivosti u gotovo svim popularnijim Web preglednicima. HTML dokumenti za manifestaciju otkrivenih ranjivosti priloženi su u arhivi `Mangleme` alata. Jedini Web preglednik otporan na `Mangleme` testove inicijalno je bio `Internet Explorer`, no nakon što je `Mangleme` alat dorađen od strane prije spomenutog člana iz Felinemenace grupe pod nadimkom Ned, otkriven je i `IFRAME` preljev spremnika u `Internet Explorer` Web pregledniku. U nastavku je prikazano postavljanje `Mangleme` alata.

```
root:~/mangleme-1.2# cp mangle.cgi /var/www/cgi-bin/
root:~/mangleme-1.2# chmod +x /var/www/cgi-bin/mangle.cgi
```

Samo korištenje navedenog alata prilično je jednostavno i izvodi se pozicioniranjem Web preglednika na URL lokaciju na kojoj je postavljena `Mangleme` CGI skripta. Web stranica se sama osvježava nakon svakog novog HTTP zahtjeva i na taj način Web preglednik automatski dobiva novi HTML dokument. Prilikom procesa *fuzzinga* Web preglednika, često se dogodi da Web preglednik zbog neobičnih HTML oznaka alocira veliki dio radne memorije i troši znatno procesorsko vrijeme što uvelike može usporiti rad računala. U takvim slučajevima ponekad je potrebno nasilno prekinuti, odnosno "ubiti" proces i krenuti ispočetka sa procesom *fuzzinga*. Iako se takvo ponašanje Web preglednika također može smatrati ranjivošću uskraćivanjem računalnih resursa (engl. ***D**enial **o**f **S**ervice*), u slučaju Web preglednika, takve ranjivosti uglavnom nisu zanimljive. Za detaljnu analizu Web preglednika potrebno je na proces spojiti debuger (program za otkrivanje i otklanjanje grešaka u programskom kodu), pomoću kojeg je moguće uloviti iznimke (engl. *exception*) uzrokovane nekim oblikom preljeva spremnika ili nekim drugim načinom nedozvoljenog prepisivanja memorije. U suprotnom se može dogoditi da Web preglednik ima definiran rukovatelj signala (engl. *signal handler*), ili rukovatelj iznimkama (engl. *exception handler* odnosno SEH), pa se prilikom nedozvoljenog pristupa memoriji uzrokovanog nekim memorijskim propustom sama ranjivost vidljivo ne manifestira i ostane neotkrivena. Na slici 1 je prikazan *fuzzing* `Mozilla` Web preglednika na Linux operacijskom sustavu. Kao što je vidljivo sa slike, HTML kod sadrži mnoštvo nestandardnih i pogrešno formatiranih HTML oznaka, koje bi mogle izazvati manifestaciju nekog sigurnosnog propusta. U slučaju da Web preglednik prilikom procesa *fuzzinga* nasilno prekine sa izvršavanjem (eng. *crash*), ili se jednostavno "zamrzne", moguće je reproducirati ranjivost analizom log datoteke (odnosno zadnjeg zabilježenog HTTP zahtjeva) i upotrebom `remangle.cgi` CGI skripte.



Slika 1: *Fuzzing* Mozilla Web preglednika

3. Fuzzing klijenata elektroničke pošte

Iako puno stariji od WWW servisa, elektronička pošta odnosno e-mail je i dalje jedan od najkorištenijih Internet servisa. Sam princip pošiljanja i primanja elektroničke pošte vrlo je jednostavan i bazira se na SMTP (engl. ***S****imple* ***M****ail* ***T****ransfer* ***P****rotocol*), odnosno MTA poslužitelju (engl. ***M****ail* ***T****ransport* ***A****gent*) i mail klijentu odnosno MUA agentu (engl. ***M****ail* ***U****ser* ***A****gent*). Komunikacija između poslužitelja odvija se SMTP protokolom. Nakon što poruka elektroničke pošte stigne na odredište, pohranjuje se u tzv. *mail spool* i tu je pohranjena dok je korisnik ne dohvati nekim od protokola za pregledavanje elektroničke pošte. Za dohvat elektroničke pošte sa *mail* poslužitelja, ukoliko korisnik nema lokalni pristup poslužitelju, koristi se POP3 (engl. ***P****ost* ***O****ffice* ***P****rotocol*) ili IMAP (engl. ***I****nternet* ***M****essage* ***A****ccess* ***P****rotocol*) protokol. Osnovni RFC 822 (engl. ***R****equest* ***F****or* ***C****omment*) dokument koji je opisivao format poruka elektroničke pošte bio je vrlo jednostavan i neadekvatan za današnje potrebe. To je jedan od osnovnih razloga zašto su RFC 2822 dokumentom definirane dodatne opcije u zaglavlju poruka elektroničke pošte te MIME (engl. ***M****ultipurpose* ***I****nternet* ***M****ail* ***E****xtensions*) ekstenzije definirane dokumentom RFC 2045 koje proširuju osnovni standard poruka i omogućavaju definiranje prijenosa različitih tipova sadržaja. Za prikaz poruke krajnjem korisniku, klijent elektroničke pošte interpretira zaglavlje poruke i ovisno o postavkama prikazuje ga korisniku. S obzirom da većinu zaglavlja poruke definira pošiljatelj poruke, ovo predstavlja iznimno pogodno mjesto za identifikaciju i analizu ranjivosti. S obzirom da klijent elektroničke pošte može sadržavati ranjivost u dijelu koda za interpretiranje zaglavlja primljene poruke, primjena *fuzzing* tehnike u ovom slučaju je iznimno praktična. U nastavku je prikazan izgled zaglavlja uobičajene poruke elektroničke pošte.

```
From: test@localhost
To: root@localhost
Subject: test poruka
Message-ID: <Pine.LNX.4.62.0509031618110.4425@t-rex.el8>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII; format=flowed
```

Kao što je vidljivo iz priloženog zaglavlja poruke, u zaglavlju se nalazi adresa pošiljatelja, adresa primatelja, tema odnosno subjekt poruke te dodatna zaglavlja koja definiraju samu poruku. Velik broj mogućih elemenata i kombinacija zaglavlja poruka elektroničke pošte povećava mogućnost postojanja sigurnosnog propusta u dijelu programskog koda koji interpretira zaglavlja. Iz tog razloga zaglavlje poruke predstavlja vrlo zanimljivo područje kada se govori o identifikaciji sigurnosnih propusta unutar klijenata elektroničke pošte. *Fuzzer* program za poruke elektroničke pošte najlakše je realizirati u

obliku jednostavnog mail klijenta koji se spaja na SMTP poslužitelj i korisniku šalje poruke sa različitim kombinacijama zaglavlja. Nakon što su testne poruke generirane i proslijeđene na željenu adresu, potrebno je mail klijentom dohvatiti poruke sa POP3, odnosno IMAP poslužitelja. Manifestacija sigurnosnog propusta može se dogoditi uglavnom prilikom samog dohvata poruke sa poslužitelja ili prilikom pregledavanja poruke, ovisno o tipu ranjivosti. Postupak pregledavanja pristiglih poruka varira između različitih klijenata elektroničke pošte, no uglavnom je držanjem tipke za pregled sljedeće poruke moguće u kratkom vremenu pregledati sve pristigle poruke.

U dodatku A priložen je jednostavan mail *fuzzer* koji korisniku s adresom `leon@localhost` šalje velik broj poruka sa različitim kombinacijama zaglavlja. U priloženoj verziji, program testira klijenta elektroničke pošte na 30 različitih elemenata zaglavlja, s različitim vrijednostima i različitom dužinom. U nastavku je prikazano pokretanje demo *fuzzer* programa.

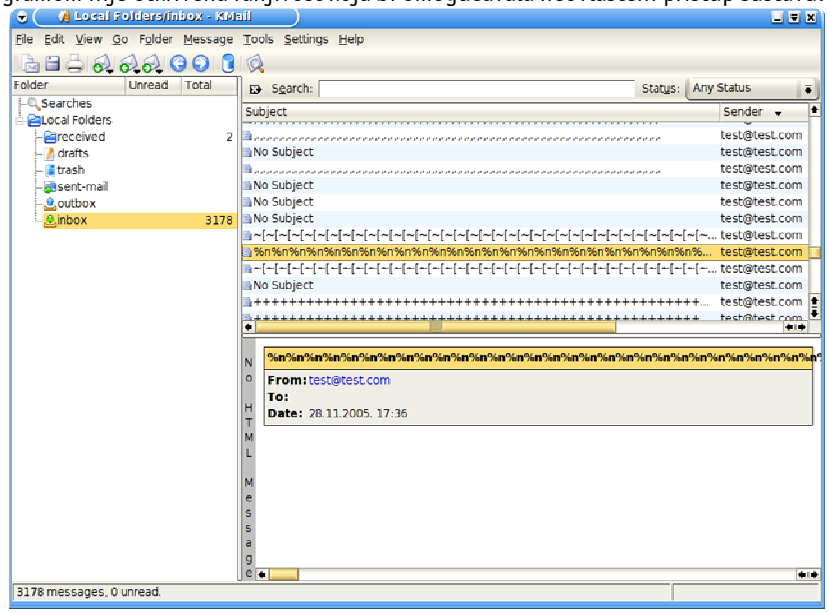
```
root@~/PROJECTS/MAILCLIENTFUZZER#./a.out 127.0.0.1
Fuzzing Date:...
Fuzzing Sender:...
Fuzzing Apparently-To:...
Fuzzing Bcc:...
Fuzzing Cc:...
Fuzzing Comments:...
Fuzzing Content-Transfer-Encoding:...
Fuzzing Content-Type:...
Fuzzing Errors-To:...
Fuzzing Expires:...
Fuzzing From:...
Fuzzing Message-Id:...
Fuzzing In-Reply-To:...
Fuzzing Mime-Version:...
Fuzzing Newsgroups:...
Fuzzing Organization:...
Fuzzing Priority:...
Fuzzing Received:...
Fuzzing References:...
Fuzzing Reply-To:...
Fuzzing Subject:...
Fuzzing To:...
Fuzzing X-Confirm-Reading-To:...
Fuzzing X-Distribution:...
Fuzzing X-Errors-To:...
Fuzzing X-Mailer:...
Fuzzing X-PMFLAGS:...
Fuzzing X-Priority:...
Fuzzing X-Sender:...
Fuzzing X-UIDL:...
root@t-rex:~/PROJECTS/MAILCLIENTFUZZER#
```

Kao što je već navedeno, nakon što su poruke poslane, potrebno ih pregledati odgovarajućim klijentom kojeg se želi analizirati. U nastavku je prikazano pokretanje popularnog mail klijenta pod nazivom `elm` nakon pošiljanja poruka koje je generirao *fuzzer* program.

```
leon@t-rex:~$ gdb elm
...
(gdb) r
Reading in /var/spool/mail/leon, message: 481
1027
1313
Program received signal SIGSEGV, Segmentation fault.
0xffffffff in ?? ()
(gdb) i r
eax            0x40189978       1075353976
ecx            0x40191268       1075384936
edx            0xffffffff       -1
ebx            0xffffffff       -1
esp            0xbfffe4a0       0xbfffe4a0
ebp            0xffffffff       0xffffffff
esi            0x80cbb21        135052065
```

```
edi            0x807ee81        134737537
eip            0xffffffff       0xffffffff
eflags         0x210246 2163270
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x0      0
(gdb) quit
The program is running.  Exit anyway? (y or n) y
leon@t-rex:~$
```

Kao što je vidljivo iz primjera, `elm` program je nasilno prekinuo izvršavanje zbog pokušaja izvršavanja instrukcija sa adrese `0xffffffff`, što upućuje na klasični preljev spremnika na stogu (engl. *stack overflow*). U ovom slučaju radi se o nedavno otkrivenom preljevu spremnika koji se javlja u slučaju kada `elm` program dohvati poruku sa elementom zaglavlja "`Expires`" koji kao vrijednost ima predugačak niz znakova. Spomenuta ranjivost je uklonjena u novijim inačicama `elm` programskog paketa, u ovom primjeru je iskorištena isključivo u svrhu demonstracije postupka *fuzzinga* na klijentima za pregledavanje elektroničke pošte. Na slici 2 prikazan je proces *fuzzinga* `KMAIL` programskog paketa, popularnog grafičkog mail klijenta za Linux operacijske sustave. U ovom slučaju, *fuzzer* programom nije otkrivena ranjivost koja bi omogućavala neovlašteni pristup sustavu.



Slika 2: *Fuzzing* `KMAIL` mail klijenta

4. Fuzzing klijenata elektroničke pošte preko POP3 poslužitelja

Svaki podatkovni ulaz aplikacije na koji dolaze podaci čiji izvor nije pouzdan predstavlja potencijalni sigurnosni rizik za korisnika te aplikacije. Prilikom procesa dohvata poruka elektroničke pošte sa POP3 ili IMAP poslužitelja, korisnik se mora pomoću mail klijenta povezati na poslužitelj, autenticirati se te željenim protokolom dohvatiti poruke. U nastavku je prikazan postupak izravnog spajanja na POP3 poslužitelj pomoću `NetCat` alata, te dohvat nove poruke elektroničke pošte.

```
root@t-rex:~/PROJECTS/MAILCLIENTFUZZER# nc 0 110
+OK
USER leon
+OK
PASS lozinka
+OK
STAT
+OK 1 615
```

```
LIST
+OK
1 615
.
RETR 1
+OK
Return-Path: <root@t-rex.el8>
Received: from t-rex.el8 (localhost [127.0.0.1])
        by t-rex.el8 (8.13.3/8.13.3) with ESMTP id jA50bK8Y004464
        for <leon@t-rex.el8>; Sat, 20 Sep 2005 01:37:20 +0100
Date: Sat, 20 Sep 2005 01:37:20 +0100 (CET)
From: # <root@t-rex.el8>
To: leon@t-rex.el8
Subject: proba
Message-ID: <Pine.LNX.4.62.0511050137130.4460@t-rex.el8>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII; format=flowed


PROBA.
.
```

Označane linije predstavljaju podatke odnosno naredbe koje klijent šalje poslužitelju, a neoznačene predstavljaju odgovor poslužitelja. Iz samog primjera je vidljivo da je POP3 protokol prilično jednostavan. Korisnik se autenticira pomoću naredbi USER i PASS, naredbom STAT otkriva broj novih poruka koje se nakon toga pregledavaju naredbom LIST i konačno slijedi sam dohvat poruke naredbom RETR. Svaki odgovor poslužitelja sadrži prefiks "+OK" što označava uspješno izvršenu operaciju ili "-ERR" što označava pogrešku. Elementi pogodni za postupak *fuzzinga* u ovom slučaju su podaci koje POP3 poslužitelj prosljeđuje klijentu elektroničke pošte kao odgovore. Znači, cilj je realizirati jednostavan POP3 poslužitelj koji će se ponašati kao *fuzzer* mail klijenta. Takav jednostavan *fuzzer* program priložen je u dodatku B. *Fuzzer* u inačici koja je priložena ima mogućnosti *fuzzinga* odgovora osnovnih POP3 naredbi, a sama upotreba prilično je jednostavna i odnosi se na postavljanje *fuzzera* na TCP port (uglavnom 110), pokretanje klijenta koji se želi analizirati, ponavljanje dohvata poruka sa pokrenutog POP3 poslužitelja dok se ne manifestira mogući sigurnosni propust. Važno je napomenuti da je *fuzzer* napisan za Windows operacijske sustave, no izbacivanjem *Winsocket*-a i dodavanjem odgovarajućih BSD *socket* zaglavlja, radi i na Linux sustavima. U nastavku je prikazano pokretanje *fuzzer* programa te povezivanje klijentske aplikacije na POP3 poslužitelj.

```
C:\Documents and Settings\Leon\My Documents\CTEST\Debug>pop324
STARTING FUZZ!!!.........
--------------------------
POP3 FUZZER.....
Coded by Leon Juranic <ljuranic@lss.hr>

RECEIVED: USER leon

RECEIVED: PASS ovo_je_samo_proba

RECEIVED: STAT

o [STAT] Sending response: +OK 3 2
 nRECEIVED: LIST

o [LIST] Sending response +OK 40944 ,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$
~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+
$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;
+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:
;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\
:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!
\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/
!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,
/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~
,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$
~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+
$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;
+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:
;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\
```

```
:;+$~,/!\:;+$~,/!\:;+$~
1 81890
2 163782
3 327566
4 655134
5 1310270
6 2620542
7 5241086
8 10482174
9 20964350
10 41928702
11 83857406
12 167714814
13 335429630
14 670859262
15 1341718526
16 -1611530242
17 1071906814
.

RECEIVED: RETR 1

o [RETR] Sending response +OK 4 ,/!\:;+$~,/!\:
o CLIENT GONE!!!...waiting for new one
```

Sa klientske strane to izgleda ovako.

```
+OK %%n FUZZ %%n accepted %%n no problem
USER leon
+OK %%n FUZZ %%n accepted %%n no problem
PASS ovo_je_samo_proba
+OK %%n FUZZ %%n accepted %%n no problem
STAT
+OK 3 2
LIST
+OK 40944 ,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+
$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;
+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:
;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\
:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!
\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/
!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,
/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$
~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+
$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;
+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:
;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\
:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~
1 81890
2 163782
3 327566
4 655134
5 1310270
6 2620542
7 5241086
8 10482174
9 20964350
10 41928702
11 83857406
12 167714814
13 335429630
14 670859262
15 1341718526
16 -1611530242
17 1071906814
.
RETR 1
+OK 4 ,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/
!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,
/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~
,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$
~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+
$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;
+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:
;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\
:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~
```

```
;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\
:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!
\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/
!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,
/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~
,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$
~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$~,/!\:;+$
```

5. Zaključak

Fuzzing tehnike sigurnosnim stručnjacima daju mogućnost da u relativno kratkom vremenskom razdoblju analiziraju sigurnost aplikacija pomoću različitih ulaznih podataka. Na taj način znatno se skraćuje vrijeme potrebno za otkrivanje sigurnosnih propusta, što može znatno povećavati učinkovitost samog postupka. Vrlo jednostavni *fuzzer* programi ukoliko su dobro napisani mogu rezultirati otkrivanjem vrlo opasnih ranjivosti, čijim se uklanjanjem znatno može podići razina sigurnosti pojedinih programa. Također, pri izradi samog *fuzzera*, logičke greške u samoj *fuzzer* aplikaciji su čak i dobrodošle, zato jer mogu rezultirati otkrivanjem nekog nestandardnog sigurnosnog propusta, koji bi inače mogao proći nezabilježeno.

6. Reference

[1] Jack Koziol, „Shellcoder's Handbook“
[2] Michal Zalewski, http://lcamtuf.coredump.cx/
[3] SMUDGE fuzzer, http://www.felinemenace.org/~nd/

Dodatak A

```
/*

Simple mail client fuzzer

Coded by Leon Juranic, ljuranic@lss.hr
LSS Security / http://security.lss.hr/

*/

#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>


char fuzzit[][64]={
"Date:",
"Sender:",
"Apparently-To:",
"Bcc:",
"Cc:",
"Comments:",
"Content-Transfer-Encoding:",
"Content-Type:",
"Errors-To:",
"Expires:",
"From:",
"Message-Id:",
"In-Reply-To:",
"Mime-Version:",
"Newsgroups:",
"Organization:",
"Priority:",
"Received:",
"References:",
"Reply-To:",
"Subject:",
"To:",
"X-Confirm-Reading-To:",
"X-Distribution:",
"X-Errors-To:",
"X-Mailer:",
"X-PMFLAGS:",
"X-Priority:",
"X-Sender:",
"X-UIDL:"};


#define GNF_FUZZ_REPEAT        1

struct mail_fuzz {
        char fzzbuf[512];
        int type;
} mail_fuzz [] = {
        {"?*<",GNF_FUZZ_REPEAT},
        {"@,",GNF_FUZZ_REPEAT},
        {";(>",GNF_FUZZ_REPEAT},
        {"&;",GNF_FUZZ_REPEAT},
        {"!!##@",GNF_FUZZ_REPEAT},
        {"\\",GNF_FUZZ_REPEAT},
        {"%n",GNF_FUZZ_REPEAT},
        {"??ABCD",GNF_FUZZ_REPEAT},
        {"(~{(",GNF_FUZZ_REPEAT},
        {"(/\\",GNF_FUZZ_REPEAT},
        {":/)",GNF_FUZZ_REPEAT},
        {":A)",GNF_FUZZ_REPEAT},
        {"A/()",GNF_FUZZ_REPEAT},
        {"\"A>!",GNF_FUZZ_REPEAT},
```

```
        {"~/)))",GNF_FUZZ_REPEAT},
        {"/)%n",GNF_FUZZ_REPEAT},
        {"$!({]\\",GNF_FUZZ_REPEAT},
        {",@<,@.>",GNF_FUZZ_REPEAT},
        {"~[",GNF_FUZZ_REPEAT},
        {"+>)",GNF_FUZZ_REPEAT},
        {"t@;@../",GNF_FUZZ_REPEAT},
        {"]}>",GNF_FUZZ_REPEAT},
        {"_{%%p",GNF_FUZZ_REPEAT},
        {";a@l.",GNF_FUZZ_REPEAT},
        {"!@,]}",GNF_FUZZ_REPEAT},
        {"$A#$((@",GNF_FUZZ_REPEAT},
        {"\\:;/*",GNF_FUZZ_REPEAT},
        {"=A;;);)",GNF_FUZZ_REPEAT},
        {"\r\n;)",GNF_FUZZ_REPEAT},
        {"\xff\xaa",GNF_FUZZ_REPEAT}};

char   SMTP[]   =   "HELO   localhost\r\nMAIL   FROM:   leon@localhost\r\nRCPT   TO:
leon@localhost\r\nDATA\r\n";


int bufsize[] = {140, 600, 1100, 4500, 10000};


char *gimme_fuzz (void)
{

        static int fzz_str=0,fzz_siz=0,cnt=0;
        static char buf[90000];
        int x;

         if (fzz_str == (sizeof(mail_fuzz)/sizeof(struct mail_fuzz)) && fzz_siz ==
sizeof(bufsize)/sizeof(int))
                 return NULL;

        memset (buf,0,sizeof(buf));
        for (x=0;x<(bufsize[fzz_siz]/strlen(mail_fuzz[fzz_str].fzzbuf));x++) {
                strcat (buf,mail_fuzz[fzz_str].fzzbuf);
        }

        if (fzz_siz == sizeof(bufsize)/sizeof(int))
        {
                fzz_str++;
                fzz_siz=0;
        }

        if (fzz_str == (sizeof(mail_fuzz)/sizeof(struct mail_fuzz)))
                fzz_str = 0;
        else
                fzz_siz++;

        {
                char buf32[100000];
                FILE *out = fopen ("/tmp/report","a");
                fprintf (out, "%d: %.15s\n",cnt++,buf);
                fclose (out);
        }

        return buf;
}



void send_data (char *arg, char *data)
{
        int sock;
        struct sockaddr_in sin;
        sock = socket (AF_INET, SOCK_STREAM, 0);

        sin.sin_addr.s_addr = inet_addr(arg);
        sin.sin_port = htons(25);
        sin.sin_family = AF_INET;
        bzero (sin.sin_zero,8);
```

```
        if (connect (sock, (struct sockaddr*)&sin, sizeof(struct sockaddr)) == -1)
        {
                perror ("connect");
                exit(-1);
        }


//      printf ("Data to send: %s\n",data);
        if (send (sock,data,strlen(data),0) == -1)
        {
                perror ("send");
                exit(-1);
        }
        usleep(5000);
        char re[1024];

        recv (sock,data,sizeof(re),0);
//      printf ("%s\n",re);

        close (sock);

}


main (int argc, char **argv)
{
        char buf[80000];
        int x,y,z;

        if (argc < 2)
        {
                printf ("Usage: %s <IP>\n",argv[0]);
                exit(-1);
        }


        for (x=0;x<sizeof(fuzzit)/sizeof(fuzzit[0]);x++)
        {
                printf ("Fuzzing %s...\n",fuzzit[x]);
                for (y=0;y<sizeof(bufsize)/sizeof(int);y++) {
                        for (z=0;z<sizeof(mail_fuzz)/sizeof(struct mail_fuzz);z++) {
                        snprintf      (buf,sizeof(buf),"%sFrom:      test@test.com\n%s
%s\n.\n", SMTP, fuzzit[x], gimme_fuzz());
                        send_data (argv[1], buf);
                        }
                }
        }

}
```

Dodatak B

```
/*
        POP3 CLIENT FUZZER
        Coded by Leon Juranic <ljuranic@lss.hr>


*/


#include <stdio.h>
#include <windows.h>


#pragma comment (lib,"ws2_32")


static int msgnum=3;  // number of messges in mailbox



char fzzstr[][16] = { ",/!\\:;+$~","\xff","-ERR ","+OK","AAAAAA" ,"%n"};


int sizes[] = {111,1110,3000,4000};


char *fuzzstr (int off, int size)
{

        char *buf;
        int x;
//      printf ("SIZE: %d\n",size);

        buf = (char*)malloc (size+6);
        memset (buf,0,size);

        for (x=0;x<(size/strlen(fzzstr[off]));x++)
                strcat (buf,fzzstr[off]);
        buf[size] = '\0';

        return buf;

}



char * cmd_STAT(void)
{
        static int off=1,size=0;
        char *buf = (char *)malloc(1024+1);

        off++;
        size ++;


        if (off == 0) off = 1;
//      if (size == 0) size = 1;

        memset (buf,0,1024);
        _snprintf (buf,1024,"+OK %u %u\r\n",off,size);
        //printf ("DEBUG: STAT: %s\n",buf);

        return buf;

//      if (size + 1 < sizeof(sizes)/sizeof(size[0])
//              size++;
//      else size=0;
}
```

```
char * cmd_RETR (void)
{
        static int off=0, size=1, fzzsize=0;
        static char *response;
        char *ret;


        response = fuzzstr(off,sizes[fzzsize]);

        ret = (char*)malloc(strlen(response)+62+3);

        memset (ret,0,strlen(response)+62);

        size *= 2;
        if (size == 0) size = 1;

        _snprintf                (ret,strlen(response)+62-1,"+OK                %d
%s\r\n%%n%%n%%nAAA%%n%%n\r\n.\r\n",size, response);


        free(response);


        if (fzzsize < sizeof(sizes)/sizeof(sizes[0])-1)
                fzzsize++;
        else
        {
                fzzsize = 0;
                if (off < sizeof(fzzstr)/sizeof(fzzstr[0]))
                        off++;
                else off=0;
        }

        return ret;

}



char * cmd_LIST (int msgnum)
{
        static int off=0, size=1, fzzsize=0,stat=1;
        static char *response, *list;
        char *ret;
        int msgsize=25,x,y;

        response = fuzzstr(off,sizes[fzzsize]);

        ret = (char*)malloc(strlen(response)+62+3);


        memset (ret,0,strlen(response)+32);

        size *= 8;
        if (size == 0) size = 1;

        stat += 8; // uvecavanje broja izlistanih poruka

        if (msgnum == 0) {
                _snprintf (ret,strlen(response)+32-1,"+OK %d %s\r\n",size, response);

                list  = (char*)malloc((msgsize*stat)+20);
                memset (list,0,(msgsize*stat)+20);

                for (x=0;x<stat;x++)
                {
                        char buf[25];
                        memset (buf,0,sizeof(buf));
                        size = (size+1) * 2;
                        _snprintf (buf,sizeof(buf),"%d %d\r\n",x+1,size);
                        strcat (list,buf);
                }
```

```
                ret = (char*)realloc (ret, strlen(ret)+strlen(list)+20);
                strcat (ret,list);
                strcat (ret,".\r\n");

                free (list);

        }

    if (stat > 10000)
        stat = 1;


        free(response);


        if (fzzsize < sizeof(sizes)/sizeof(sizes[0])-1)
                fzzsize++;
        else
        {
                fzzsize = 0;
                if (off < sizeof(fzzstr)/sizeof(fzzstr[0]))
                        off++;
                else off=0;
        }

        return ret;

}




char * cmd_QUIT (void)
{
        static int off=0, size=1, fzzsize=0;
        static char *response;
        char *ret;


        response = fuzzstr(off,sizes[fzzsize]);

        ret = (char*)malloc(strlen(response)+32+3);

        memset (ret,0,strlen(response)+32);

        _snprintf (ret,strlen(response)+32-1,"+OK %s\r\n",response);

        free(response);


        if (fzzsize < sizeof(sizes)/sizeof(sizes[0])-1)
                fzzsize++;
        else
        {
                fzzsize = 0;
                if (off < sizeof(fzzstr)/sizeof(fzzstr[0]))
                        off++;
                else off=0;
        }

        return ret;

}


char * cmd_DELE (void)
{
        static int off=0, size=1, fzzsize=0;
        static char *response;
        char *ret;
```

```
        response = fuzzstr(off,sizes[fzzsize]);

        ret = (char*)malloc(strlen(response)+32+3);

        memset (ret,0,strlen(response)+32);

        _snprintf (ret,strlen(response)+32-1,"+OK %s\r\n",response);

        free(response);


        if (fzzsize < sizeof(sizes)/sizeof(sizes[0])-1)
                fzzsize++;
        else
        {
                fzzsize = 0;
                if (off < sizeof(fzzstr)/sizeof(fzzstr[0]))
                        off++;
                else off=0;
        }

        return ret;

}



int handle_client (int sock, struct sockaddr *sin)
{

        char buf[5000],tmp[5000], *response;
        fd_set fds;
        int n;
        char auth[]="+OK %%n FUZZ %%n accepted %%n no problem\r\n";
        struct timeval tv;

        tv.tv_sec = 2;
        tv.tv_usec = 0;

        send (sock,auth,strlen(auth),0);


        FD_ZERO(&fds);
        FD_SET (sock,&fds);

        while (1)
        {
                if (select (NULL,&fds,NULL,NULL,NULL) == -1)
                {
                        printf ("SELECT :(((\n");
                        return 0;
                }

                if (FD_ISSET(sock,&fds))
                {

                        memset (buf,0,sizeof(buf));
                        if ((n = recv (sock, buf, sizeof(buf),0)) <= 0)
                                return 0;
                        buf[n] = '\0';
                        printf ("RECEIVED: %s\n",buf);
                }

                if (strstr (buf,"USER") != NULL)
                        send (sock, auth, strlen(auth),0);

                else if (strstr (buf,"PASS") != NULL)
                        send (sock, auth, strlen(auth),0);


                else if (strstr (buf,"STAT") != NULL) {
                        response = cmd_STAT();
```

```
                printf ("o [STAT] Sending response: %.20s n",response);
                send (sock,response,strlen(response),0);
                free(response);

                }

        else if (strstr (buf,"RETR") != NULL)
        {
                response = cmd_RETR();
                printf ("o [RETR] Sending response %.20s\n",response);
                {
                        send (sock,response,strlen(response),0);
                }
                        free(response);

        }


        else if (strstr (buf,"LIST") != NULL)
        {
                response = cmd_LIST(0);
                printf ("o [LIST] Sending response %s\n",response);
                send (sock,response,strlen(response),0);
                free(response);
        }

        else if (strstr (buf,"DELE") != NULL)
        {
                response = cmd_DELE();
                printf ("o [DELE] Sending response %.20s\n",response);
                {

                        send (sock,response,strlen(response),0);
                }
                free(response);
        }


        else if (strstr (buf,"NOOP") != NULL)
        {
                response = cmd_QUIT();
                printf ("o [NOOP] Sending response %.20s\n",response);
                send (sock,response,strlen(response),0);
                free(response);
        }

        else if (strstr (buf,"RSET") != NULL)
        {
                response = cmd_QUIT();
                printf ("o [RSET] Sending response %.20s\n",response);
                send (sock,response,strlen(response),0);
                free(response);
        }

        else if (strstr (buf,"UIDL") != NULL);

        else if (strstr (buf,"AOP") != NULL);

        else if (strstr (buf,"QUIT") != NULL)
        {
                response = cmd_QUIT();
                printf ("o [QUIT] Sending response %.20s\n",response);
                send (sock,response,strlen(response),0);
                free(response);
                return 0;
        }


        else printf ("ELSE!!!!!!!!!!!!!!!!!!!!!!!!!!!\n");;

}
```

```
}


int POP3_server(void)
{
        struct sockaddr_in sin,cli;
        int sock,clsock;
        WORD wVersionRequested;
        WSADATA wsaData;
        int err;


        wVersionRequested = MAKEWORD( 2, 2 );
        err = WSAStartup( wVersionRequested, &wsaData );
        if ( err != 0 ) {
                printf ("ERROR: Sorry, cannot create socket!!!\n");
                exit(-1);
        }


        sock = socket (AF_INET, SOCK_STREAM,0);

        sin.sin_family = AF_INET;
        sin.sin_port = htons (110);
        sin.sin_addr.s_addr = INADDR_ANY;
        memset (sin.sin_zero,0,8);

        DWORD optval = 1;
        setsockopt                              (sock,SOL_SOCKET,SO_REUSEADDR,(const
char*)&optval,sizeof(optval));

        if (bind(sock,(struct sockaddr*)&sin,sizeof(sockaddr)) == -1)
        {
                printf ("CANNOT BIND TO PORT!!!!!\n");
                exit(-1);
        }


        listen (sock, 10);


        int three = sizeof(struct sockaddr);

        while (clsock = accept (sock,(struct sockaddr*)&cli,&three))
        {
                handle_client(clsock,(struct sockaddr*)&cli);
                printf ("o CLIENT GONE!!!...waiting for new one\n");
                closesocket (clsock);
        }

        closesocket (sock);


        return 0;
}



void main (int argc, char **argv)
{

        printf ("STARTING FUZZ!!!.........\n"
                        "-------------------------\n");
        printf ("POP3 FUZZER.....\n"
                        "Coded by Leon Juranic <ljuranic@lss.hr>\n");
        POP3_server();
}
```