Otkrivanje sigurnosnih propusta fuzzing tehnikom

INFIGO-TD-2006-04-01

2006-04-25

Leon Juranić
Leon.Juranic@infigo.hr

INFIGO Information security

Infigo IS d.o.o.
Horvatovac 20
10000 Zagreb

tel. +385 1 4662 700
fax. +385 1 4662 701
info@infigo.hr
www.infigo.hr

INFIGO
Information security

SADRŽAJ

1. UVOD

Najveći dio provala u računalne sustave događa se kao posljedica ranjivosti u aplikativnim i poslužiteljskim programskim paketima. Otkrivanje i identifikacija sigurnosnih propusta interesantan je proces za sigurnosne stručnjake i administratore računalnih sustava, ali i za neovlaštene korisnike koji žele penetrirati u tuđe računalne sustave. Otkrivanjem novih sigurnosnih propusta i izradom specijaliziranih programa za njihovo iskorištavanje, moguće je ostvariti neovlašteni pristup velikom broju računala na Internetu što predstavlja ozbiljnu prijetnju za sve korisnike informacijskih tehnologija.

Postoji nekoliko provjerenih metoda za otkrivanje sigurnosnih propusta:

- analiza izvornog koda,
- analiza binarnih datoteka (statička i dinamička (eng. *runtime*) analiza),
- *runtime* analiza API funkcija,
- *fuzzing* metoda (tzv. *fault injection*) i
- hibridne metode (razne kombinacije prije navedenih metoda)

Iako je pomoću svih navedenih metoda moguće otkriti sigurnosne propuste, neke od njih posebno su zanimljive budući da omogućavaju znatno brže i jednostavnije otkrivanje sigurnosnih propusta. Posljednjih godina, posebnu pozornost privukla je tehnika *fuzzinga*, metoda pomoću koje je relativno brzo moguće otkriti kritične sigurnosne propuste u različitim programskim rješenjima. Pod kritičnim sigurnosnim propustima uglavnom se podrazumijevaju različite varijacije preljeva spremnika (eng. *buffer overflow*), kod kojih je neovlašteni korisnik u mogućnosti prepisati kritične dijelove memorijskog prostora ranjivog programa odnosno procesa. Rezultat uspješnog iskorištavanja preljeva spremnika najčešće je izvršavanje tzv. *shellcode* programskog koda, niza specijalno prilagođenih procesorskih instrukcija koje je u proces ubacio neovlašteni korisnik s ciljem ostvarivanja neovlaštenog pristupa.

Fuzzing metoda bazira se na tzv. *fault injection* tehnici kojom se, prosljeđivanjem različitih ulaznih podataka ciljnoj aplikaciji, nastoje otkriti sigurnosni propusti. Kod *fuzzing* tehnike važno je da ulazni podaci budu izmijenjeni tako da mogu uzrokovati manifestaciju sigurnosnog propusta, a da istovremeno mogu proći inicijalnu provjeru (eng. *sanity check*) u ciljanoj aplikaciji. *Fuzzing* metode se najčešće primjenjuju na poslužiteljskim aplikacijama, klijentskim aplikacijama, parserima datoteka, SUID programima i sl. Uglavnom, svaka aplikacija koja na neki način obrađuje korisnički unos pogodna je za primjenu *fuzzing* metode otkrivanja sigurnosnih propusta.

Fuzzing tehnikama moguće je identificirati skoro sve poznate tipove sigurnosnih propusta kao što su:

- preljevi spremnika *(*eng. *buffer overflow)*,
- preljev cjelobrojne vrijednosti *(*eng. *integer overflow)*,
- *format string* ranjivosti,
- *race condition* ranjivosti,
- umetanje SQL naredbi (eng. *SQL Inject*),
- *Cross Site Scripting* – XSS,
- udaljeno izvršavanje naredbi (eng. *remote command execution*),
- napadi na datotečni sustav (*reverse traversal* i sl.),
- *information leak* ranjivosti itd.

Veliki broj sigurnosnih ranjivosti koje se u današnje vrijeme otkrivaju i objavljuju na svjetskim referentnim *mailing* listama i Web stranicama otkriveno je upravo korištenjem *fuzzing* tehnike. Također, sve je veći broj različitih *fuzzer* alata koji omogućuju ispitivanje sigurnosti različitih programskih rješenja, što također ukazuje na iznimnu popularnost ove metode.

Uz sve svoje prednosti, važno je napomenuti da *fuzzing tehnika* nije univerzalna metoda za otkrivanje ranjivosti. Ponekad je za otkrivanje pojedinih ranjivosti potrebno zadovoljiti specifične uvjete ciljane aplikacije za koje *fuzzing* alat nije prilagođen te je u takvim slučajevima potrebno primijeniti neke druge metode ovisno o tipu propusta koji se analizira. Kada se govori o mrežnim aplikacijama, važno je napomenuti da je *fuzzing* tehnika vrlo pogodna i za općenito testiranje stabilnosti aplikacije budući da ovakvi testovi mogu biti destruktivni.

2. TEHNIKE FUZZINGA

Iako pojam *fuzzinga* označava otkrivanje sigurnosnih propusta metodom slanja različitih, potencijalno malicioznih ulaznih podataka nekoj aplikaciji, postoji više različitih tehnika i metoda *fuzzinga*, ovisno o načinu generiranja ulaznih podataka. U nastavku dokumenta su opisani neki od osnovnih tipova *fuzzinga*.

2.1. FUZZING PRIKUPLJENIH PODATAKA O SJEDNICI

Ova metoda bazira se na analizi dijelova već završene legitimne sjednice protokola aplikacije koju se želi testirati, inkrementalnom mijenjanju pojedinih podatkovnih vrijednosti i prosljeđivanju izmijenjenih rezultata aplikaciji. Tehnika se bazira na činjenici da *fuzzer* mijenja već postojeće podatke i na taj način unosi mogućnost pojave nedeterminiranih stanja, odnosno manifestaciju sigurnosnih ranjivosti. U nastavku je prikazan pojednostavljeni primjer opisanog načina *fuzzinga* nad SMTP protokolom. Podaci koji se šalju mijenjaju se vrijednostima koje su već korištene u samom protokolu.

```
mail from: test@localhost
mailmailmailmailmailmailmailmail from: test@localhost
mail fromfromfromfromfromfromfrom: test@localhost
mail from:::::::::::::::::::::::::: test@localhost
mail from: testtesttesttesttesttesttesttesttest@localhost
mail from: test@@@@@@@@@@@@@@@@@@@@localhost
mail from: test@localhostlocalhostlocalhostlocalhost
```

Ova metoda *fuzzinga* prilično je jednostavna za implementaciju, a može rezultirati otkrivanjem vrlo zanimljivih ranjivosti.

2.2. SPECIJALIZIRANI FUZZERI

Drugi, vrlo popularan način *fuzzinga* je izrada specijaliziranog *fuzzer* programa koji je posebno prilagođen ispitivanju implementacije određenog protokola. Prednost takvih *fuzzer* programa je dobro poznavanje i prilagođenost protokolu koji se analizira, što im omogućuje provođenje detaljnijih i efikasnijih testova nad ciljnom aplikacijom. Izrada specijaliziranih *fuzzer* programa posebno je korisna kod protokola odnosno mrežnih aplikacija koje otvaraju sjednicu u kojoj se između klijenta i poslužitelja izmjenjuje više kontrolnih podataka (npr. SMTP, FTP, IMAP, SSH, itd.).

U takvim slučajevima *fuzzer* je obično namijenjen specifičnom protokolu i elementima protokola koje će testirati. Nedostatak ovakvih *fuzzer* programa je to što ovdje sam programer definira protokol i elemente protokola koji će se *fuzzati*, pa konačnu kvalitetu i učinkovitost *fuzzera* kao proizvoda određuje sposobnost odnosno kompetentnost programera.

2.3. GENERIČKI FUZZERI

Izraz generički *fuzzer* odnosi se na alat koji korisniku omogućava da sam opiše protokol i elemente protokola nad kojim će se primijeniti *fuzzing* metoda. Velika prednost takvih alata je što mogu relativno brzo ispitati sigurnost implementacije različitih protokola, no s druge strane, generički *fuzzeri* nisu efikasni kao specijaliziranih *fuzzeri*, te zbog toga mogu previdjeti neke sigurnosne ranjivosti. Svi testovi koje provodi *fuzzer* moraju biti definirani od strane korisnika, što kod složenijih protokola može predstavljati ograničenje. Vrlo popularan generički *fuzzer* alat je SPIKE, kojeg se može pronaći na URL adresi http://www.immunitysec.com.

Jedan od nedostataka generičkih *fuzzer* alata je taj što najčešće nisu prilagođeni manje iskusnim korisnicima.

3. FUZZING POSLUŽITELJSKIH APLIKACIJA

Neovlaštenim korisnicima najinteresantnije mete za *fuzzing* su upravo mrežne aplikacije zbog same činjenice da se koriste na Internetu i da su dostupne velikom broju korisnika. To su uglavnom mrežni servisi za popularne protokole kao što su HTTP, FTP, SMTP, POP3, SSH iako postoje i drugi manje popularni mrežni servisi koji mogu biti zanimljivi za *fuzzing*.

S obzirom da se radi o vrlo popularnim protokolima i servisima koje svakodnevno koristi velik broj korisnika, otkrivanje sigurnosnih ranjivosti u implementaciji tih protokola neovlaštenom korisniku daje veliki izbor ranjivih poslužitelja koje može kompromitirati.

Broj ranjivosti koje bi se mogle pojaviti u implementaciji nekog protokola (odnosno u poslužiteljskom softveru) eksponencijalno raste sa kompleksnošću odnosno mogućnostima samog protokola. Mogućnosti i dizajn *fuzzer* programa vrlo su važni čimbenici koji utječu na efikasnost *fuzzing* metode. Cilj je napraviti *fuzzer* program koji će sa što manje testova u razumnom vremenskom periodu na što bolji način i što detaljnije testirati sigurnost aplikacije koja se ispituje.

Kada se radi o *fuzzingu* poslužiteljske aplikacije, vrlo je bitno pokriti sve komponente protokola koji se testira, jer izostavljanje implementacije jednog naizgled nevažnog dijela protokola može rezultirati propuštanjem otkrivanja opasnih sigurnosnih ranjivosti.

Kako bi se skratilo vrijeme potrebno za generiranje svih mogućih ulaznih podataka koji mogu uzrokovati otkrivanje sigurnosnih ranjivosti, *fuzzer* programi najčešće imaju listu predefiniranih testnih podataka i njihove definirane veličine koje će slati ciljnoj aplikaciji. Lista testnih podataka su uglavnom znakovni nizovi (eng. *string*) koji se multipliciraju do definiranih veličina i šalju u ciljnu aplikaciju. Veličina podataka koji se šalju vrlo je važna za proces *fuzzinga*. Prekratak niz znakova najčešće neće izazvati otkrivanje sigurnosnog propusta, dok će predugačak niz znakova biti odbačen zbog svoje dužine i ranjivost će ostati neotkrivena. Iz tog razloga je veličine nabolje definirati u skladu s uobičajenim veličinama međuspremnika (engl. *buffer*) u aplikacijama.

Veličine međuspremnika koji se koriste u aplikacijama uglavnom su potencije broja 2, pa je prilikom definiranja testnih ulaznih vrijednosti preporučljivo koristiti veličine koje su nešto veće od veličina međuspremnika (npr. 20, 40, 70, 130, 260 i sl.). Također, vrlo je važno da podaci koji se šalju ciljnoj aplikaciji budu u korelaciji sa protokolom koji se testira. Pritom se misli na pojedine specijalne znakove i vrijednosti koje su specifične za određeni protokol, odnosno njegovu implementaciju koja se ispituje. Npr. ako se radi *fuzzingu* SMTP poslužitelja, podaci koji se šalju aplikaciji moraju imati veze sa konstrukcijom adresa elektroničke pošte, odnosno sadržavati znakove kao što su '@', ';', '.', '<', '>', i sl. Ukoliko se radi o HTTP poslužitelju, podaci moraju sadržavati elemente kao što su '%xx', '../', '?', '=' itd.

Što su testni podaci bolje prilagođeni protokolu čija se implementacija testira, veća je vjerojatnost da će sigurnosni propust biti otkriven. Također, *fuzzer* treba testirati aplikaciju sa što je moguće više različitih podataka, uključujući i znakove izvan vidljivog (alfanumeričkog, interpunkcijskog i sl.) ASCII seta. Uz predefinirane podatke i veličine u kojima će ti podaci biti slani, *fuzzer* može generirati i pseudo-slučajne podatke kojima proširuje spektar testova izvan samih testova koje je definirao programer *fuzzer* alata.

Primjena *fuzzing* metode na poslužiteljskim aplikacijama biti će demonstrirana u nastavku dokumenta na nekim poznatijim FTP poslužiteljima korištenjem *Infigo FTPStress Fuzzer* alata. Navedeni alat pokazao se vrlo uspješnim u otkrivanju kritičnih sigurnosnih ranjivosti u velikom broju komercijalnih i besplatnih FTP poslužitelja. Sam dizajn alata prilično je jednostavan i bazira se na ranije opisanim konceptima.

Testiranje sigurnosnih ranjivosti u poslužiteljskom softveru može se podijeliti na tzv. *preauth* i *postauth* ranjivosti. Kao što ime govori, *preauth* ranjivosti ne zahtijevaju prethodnu autentikaciju korisnika pa su stoga puno opasnije od *postauth* ranjivosti za koje maliciozni korisnik najčešće treba važeće korisničko ime i zaporku. U popularnim poslužiteljskim programskim paketima

preauth ranjivosti su uglavnom manje prisutne nego *postauth* ranjivosti. Dobar *fuzzer* alat trebao bi imati mogućnost testiranja cijelog protokola, uključujući i *preauth* i *postauth* faze.

U potrazi za novim ranjivostima, također je važno da *fuzzer* može detektirati već prije otkrivene, objavljene i eliminirane ranjivosti u softveru za koji je namijenjen. Na taj način moguće je direktno testirati efikasnost samog *fuzzing* alata.

4. NADGLEDANJE APLIKACIJE KOJA SE TESTIRA

Prilikom testiranja određene aplikacije *fuzzing* tehnikom, vrlo je važno da se ponašanje aplikacije nadgleda i analizira odgovarajućim alatima. Prilikom provođenja procesa *fuzzinga* sa ciljem otkrivanja ranjivosti preljeva spremnika, najvažnije je nadgledanje ponašanja procesa koji se testira. Za nadgledanje procesa odnosno programa u izvršavanju koriste se alati za otkrivanje i uklanjanje grešaka u programskom kodu, popularni *debuggeri*

Popularan i besplatan *debugger* za Windows platformu je OllyDbg (http://home.t-online.de/home/Ollydbg), dok se za Unix platforme najčešće koristi GDB (http://www.gnu.org/software/gdb) koji standardno dolazi sa većinom Unix baziranih operacijskih sustava. Ostali parametri koji se uglavnom nadgledaju prilikom procesa *fuzzinga* su korištenje radne memorije, mrežne aktivnosti, aktivnosti na datotečnom sustavu, pristup *registry* datoteci i sl. Za većinu ovih parametara postoje vrlo korisni i besplatni Windows alati, a neke od njih moguće je dohvatiti s adrese http://www.sysinternals.com.

Za nadgledanje korištenja datotečnog sustava dostupan je alat *File Monitor*, za detaljne informacije o procesima na sustavu *Process Explorer*, a za pristup *registry* datoteci *Registry Monitor*.

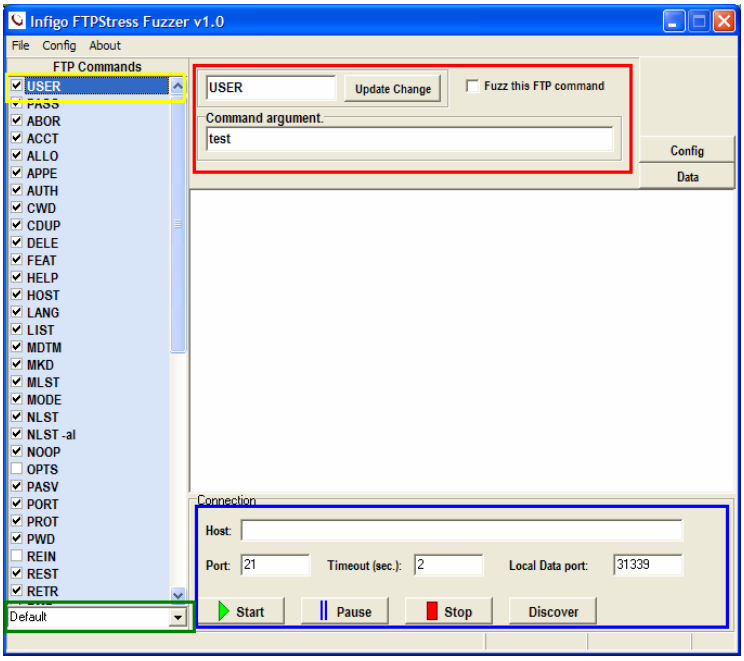
5. FUZZING FTP POSLUŽITELJA

FTP protokol relativno je jednostavan protokol, no zbog velikog broja naredbi i različitih parametara postoji mnogo podatkovnih ulaza koji mogu uzrokovati ranjivost. Ranjivosti u FTP poslužiteljima uglavnom su ranjivosti preljeva spremnika i druge memorijske ranjivosti kao što su *format string* ranjivosti, ranjivosti pri pristupu datotečnom sustavu kod kojih neovlašteni korisnik ima mogućnosti pregledavanja datoteka izvan tzv. *ftproot* direktorija itd.

Prilikom *fuzzinga* FTP poslužitelja, potrebno je testirati sve naredbe FTP protokola korištenjem što više različitih unosa. Također je potrebno ispitati pojedina stanja u koja FTP poslužitelj može doći prilikom prijenosa datoteka, prije i poslije autentikacije korisnika te drugih aktivnosti.

Jedan od alata kojim je moguće testirati sigurnost FTP poslužiteljskih aplikacija je *Infigo FTPStress Fuzzer* alat kojeg je moguće dohvatiti sa Web stranica tvrtke INFIGO IS (http://www.infigo.hr/hr/in_focus/tools). Navedeni alat korisniku omogućuje da sam definira FTP naredbe koje se žele testirati, veličinu i tip podataka koji će se slati ciljnoj aplikaciji te način provođenja *fuzzing* metode. Alat je besplatan i namijenjen je Windows operacijskim sustavima.

Na slici (Slika 1) prikazan je osnovni prozor *Infigo FTPStress Fuzzer* alata.



Slika 1: Infigo FTPStress Fuzzer

Na lijevoj strani glavnog prozora nalazi se lista osnovnih naredbi i ekstenzija FTP protokola. Unutar žutog kvadrata nalazi se FTP naredba **USER** kojom se aplikaciji prosljeđuje korisničko ime korisnika koji pristupa FTP poslužitelju. Ukoliko je pripadajuća kvačica uključena znači da će se prilikom *fuzzing* procesa označena naredba slati FTP poslužitelju.

Unutar crvenog kvadrata navedeni su dodatni parametri označene naredbe:

- ime naredbe (**USER**),
- argument koji naredba prima (**test**),
- polje **Fuzz this FTP command** kojim se označava da li će *fuzzing* metoda primijeniti na ovu naredbu. Ukoliko je potvrdni okvir (eng. *checkbox*) uključen, definirani argument naredbe se ignorira i argument naredbe se zamjenjuje *fuzzing* vrijednostima definiranim u postavkama alata. Ukoliko ovo polje nije označeno, poslužitelju se prosljeđuje argument naredbe koji je naveden unutar polja **Command argument**.

Ukoliko poslužitelj želimo testirati za *preauth* ranjivostima, **USER** i **PASS** naredbama treba uključiti **Fuzz this FTP command** potvrdni okvir. Ukoliko želimo testirati *postauth* fazu,

naredbama **USER** i **PASS** potrebno je za argumente staviti važeće korisničko ime i lozinku, te isključiti **Fuzz this FTP command** potvrdni okvir. Unutar zelenog kvadrata moguće je odabrati određene setove naredbi grupirane u predefinirane kategorije.

Unutar plavog kvadrata nalaze se postavke za mrežu odnosno odredišna IP adresa, odredišni mrežni port na kojem se nalazi FTP poslužitelj, *timeout* vrijednost i lokalni podatkovni port za prijenos podataka. *Timeout* vrijednost vrlo je bitna za proces *fuzzinga*, jer je pogrešnim odabirom *timeout* vrijednosti moguće zaobići otkrivanje neke sigurnosne ranjivosti ili znatno usporiti *fuzzing* proces. Za testiranje na lokalnom računalu, optimalna *timeout* vrijednost je od 4 do 8, ovisno o ponašanju FTP poslužitelja.

Klikom na gumb **Config** otvara se sučelje unutar kojega je moguće detaljnije podesiti način rada programa. Unutar ovog prozora moguće je precizno definirati veličinu i tip podataka koji se šalju te tip *fuzzing* metode koji će se koristiti.

Sve predefinirane vrijednosti moguće mijenjati što omogućava fleksibilnost *fuzzing* procesa. *Infigo FTPStress Fuzzer v1.0* podržava dva osnovna tipa *fuzzinga*:

- **Fuzz only one command at time** i
- **Fuzz all selected commands in same FTP session**

Ukoliko je označen prvi tip *fuzzinga* (**Fuzz only one command at time**), *fuzzer* će testirati svaku naredbu zasebno sa svakim odabranim podatkom svake odabrane veličine. U nastavku je prikazan rezultat slanja podataka ukoliko je odabran ovaj tip *fuzzinga*.

```
[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 30 ]
RECV: 226 abort successful

[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 70 ]
RECV: 226 abort successful

[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 150 ]
RECV: 226 abort successful

[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 330 ]
RECV: 226 abort successful

[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 520 ]
RECV: 226 abort successful

[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 700 ]
RECV: 226 abort successful
```

Fuzzer prvo šalje naredbu **ABOR** sa svakom definiranom veličinom znakovnog niza i sa svakim definiranim testnim podatkom.

Ukoliko je odabran drugi tip *fuzzinga* (**Fuzz all selected commands in same FTP session**), rezultat će biti kao u nastavku.

```
[ CMD: [ABOR]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 30 ]
RECV: 226 abort successful

[ CMD: [ACCT]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 30 ]
RECV: 500 Account disabled, contact administrator

[ CMD: [ALLO]    FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 30 ]
RECV: 200 allocation ok

 [ CMD: [APPE]   FUZZ: [AAAAAAAAAAAAAAAAAAAA]          SIZE: 30 ]
RECV: (null)
```

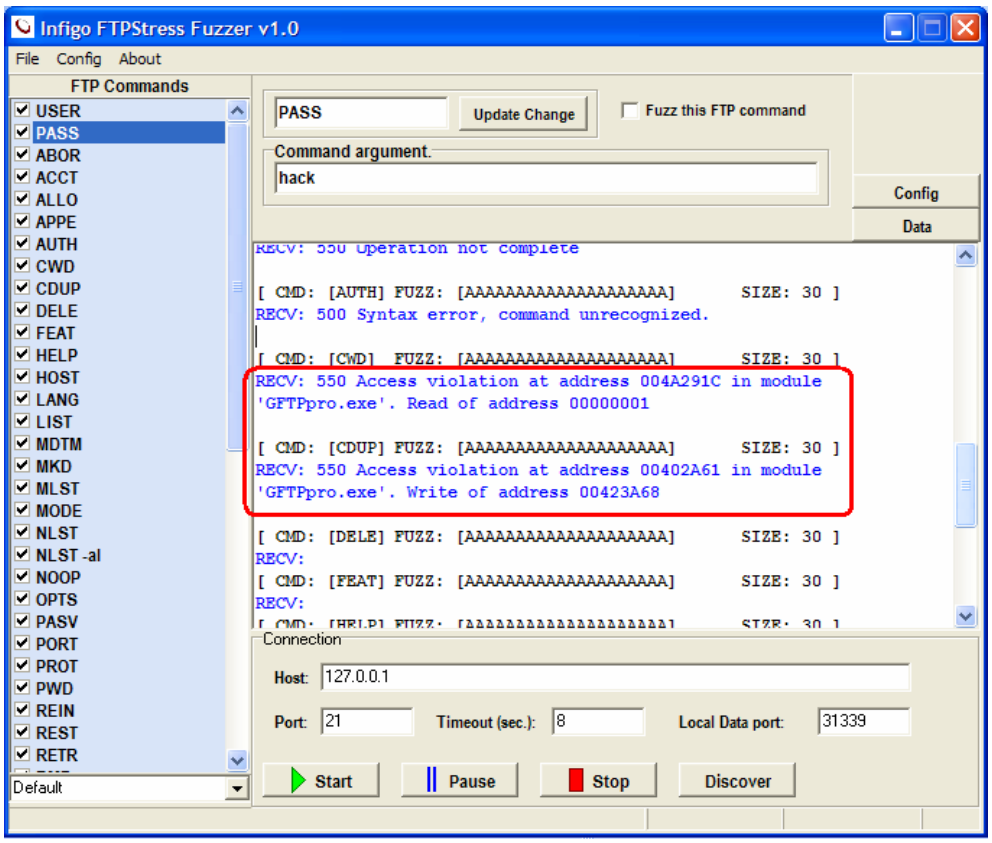
U ovom slučaju program šalje svaku odabranu naredbu s istim testnim podatkom i istom veličinom testnog podatka. Nakon testiranja svih naredbi, *Fuzzer* prelazi na iduću testnu veličinu. Različiti tipovi *fuzzing* metode mogu otkriti različite ranjivosti. Prilikom provođenja *fuzzinga* vrlo je važno analizirati ponašanje poslužitelja i prilagoditi program FTP poslužitelju koji se ispituje.

6. FUZZING STVARNIH APLIKACIJA

Fuzzing FTP poslužitelja demonstriran je na nekoliko stvarnih primjera u kojima su prikazane *0day* ranjivosti (u trenutku testiranja). Kao primjer su korišteni poslužitelji *GoldenFTPd*, *WarFTPD* i *ArgoSoft FTP Server*.

6.1. GOLDENFTPD

Golden FTP Server moguće je dohvatiti sa adrese http://www.goldenftpserver.com/ Nakon postupka instalacije, kreirano je korisničko ime '**test**' sa lozinkom '**hack**' koje je korišteno u postupku *fuzzinga*. Nakon toga pokrenut je FTP poslužitelj te *Infigo FTPStress Fuzzer* program. Kao argument naredbi **USER** navedeno je korisničko ime '**test**', a naredbi **PASS** zaporka '**hack**'. U polje '**Host**' potrebno je upisati IP adresu računala na kojem se nalazi *Golden FTP Server* poslužitelj, a u polje '**Port**' TCP port na kojem se poslužitelj nalazi. *Timeout* vrijednost u ovom je primjeru postavljena na vrijednost 8. Kao tip *fuzzinga* odabrana je **Fuzz all selected commands in same FTP session** metoda. Na slici (Slika 2) prikazano je pokretanje *Infigo FTPStress Fuzzer* programa sa opisanim postavkama.



Slika 2: Proces fuzzinga

Rezultati *fuzzinga* su primjetni već nakon prvih nekoliko testova. Naime, *GoldenFTPd* generira poruku **Access violation**, iznimku koja označava da proces pokušava čitati dio memorijskog prostora koji mu nije dostupan. Druga greška je još zanimljivija, budući da proces pokušava pisati po memoriji koja mu nije dostupna ili nije predviđena za pisanje. *GoldenFTPd* poslužitelj sigurnosnim stručnjacima ili potencijalnim napadačima čak i olakšava pronalaženje sigurnosnih propusta, pošto korištenjem rukovatelja iznimki (eng. *exception handler*) hvata iznimke i daje detaljne informacije o kakvim se iznimkama radi i zašto do njih dolazi. Ovakvo ponašanje dodatno olakšava posao iskorištavanja preljeva spremnika, budući da otkriva osjetljive informacije o memorijskom okruženju procesa. Ukoliko *Golden FTPd* sadrži ozbiljnije ranjivosti,

neovlaštenom korisniku je ovakvo ponašanje dovoljno da napiše ciljani zlonamjerni kod (eng. *exploit*), čak i bez da ima inačicu *Golden FTPd* poslužitelja.

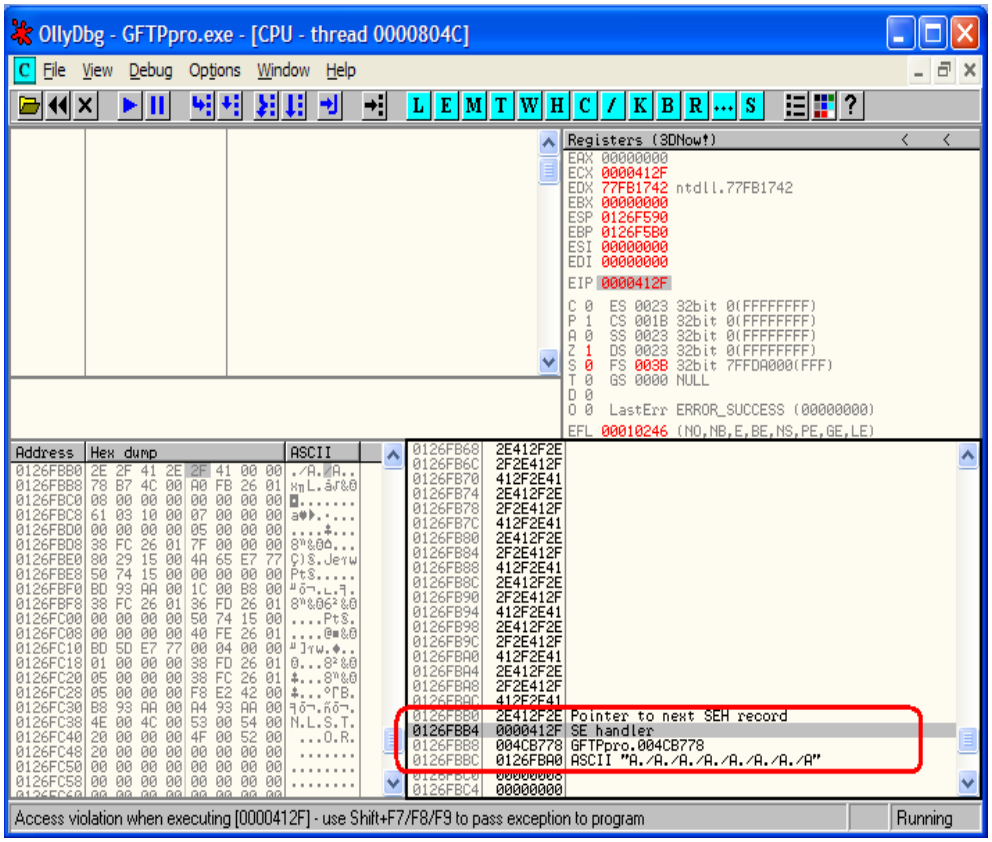
Kako bi detaljnije analizirali ranjivosti, potrebno je spojiti *OllyGdb debugger* na proces **GFTPpro.exe**. S obzirom da **Access violation** iznimaka ima jako puno, a poslužitelj ima definiran rukovatelj iznimkama, u *OllyGdb* programu potrebno je uključiti opciju da sve **Access violation** iznimke prosljeđuje aplikaciji, a aplikacija će prekinuti izvršavanje samo zbog iznimki koje nije u stanju procesirati. Za ovakve postavke potrebno je u **Options->Debugging Options->Exceptions** postaviti kvačicu na **Memory access violation** i **Single step break**. Također, kako bi se skratio proces *fuzzinga*, biti će odabrana samo osnovna grupa naredbi koje će se *fuzzati* i koja će u ovom slučaju biti podatkovne naredbe (**File Commands**).

Nakon otprilike jedne minute *fuzzinga Infigo FTPStress Fuzzer* javlja poruku da se više ne može spojiti na poslužitelj.

```
[ Connecting to 127.0.0.1:21... ]
[ Connected, starting fuzz process... ]
220 Golden FTP Server PRO ready v2.50

[ USER: [test] ]
331 User name okay, need password.

[ PASS: [hack] ]
230 User logged in, proceed.

> Sending PORT Command - PORT 127,0,0,1,122,107

RECV: 200 PORT Command successful.

[ CMD: [NLST -al] FUZZ: [./A./A./A./A./A./]  SIZE: 330 ]
[ Connecting to 127.0.0.1:21... ]
[ Connected, starting fuzz process... ]
[ USER: [test] ]
[ PASS: [hack] ]
[ CMD: [RMD]      FUZZ: [./A./A./A./A./A./]  SIZE: 330 ]
[ Connecting to 127.0.0.1:21... ]
[ Connected, starting fuzz process... ]
[ USER: [test] ]
[ PASS: [hack] ]
[ CMD: [RNFR]     FUZZ: [./A./A./A./A./A./]  SIZE: 330 ]
[ Connecting to 127.0.0.1:21... ]
[ Connected, starting fuzz process... ]
[ USER: [test] ]
[ PASS: [hack] ]
[ CMD: [RNTO]     FUZZ: [./A./A./A./A./A./]  SIZE: 330 ]
[ Connecting to 127.0.0.1:21... ]
[ Connected, starting fuzz process... ]
[ USER: [test] ]
[ PASS: [hack] ]
[ CMD: [SIZE]     FUZZ: [./A./A./A./A./A./]  SIZE: 330 ]
[ Connecting to 127.0.0.1:21... ]
[ Connected, starting fuzz process... ]
[ USER: [test] ]
[ PASS: [hack] ]
[ CMD: [XCUP]     FUZZ: [./A./A./A./A./A./]  SIZE: 330 ]
[ Connecting to 127.0.0.1:21... ]
[ ERROR: Cannot connect to target!!!         ]
[ SERVER IS MAYBE DEAD BECAUSE OF FUZZING!!! ]
```

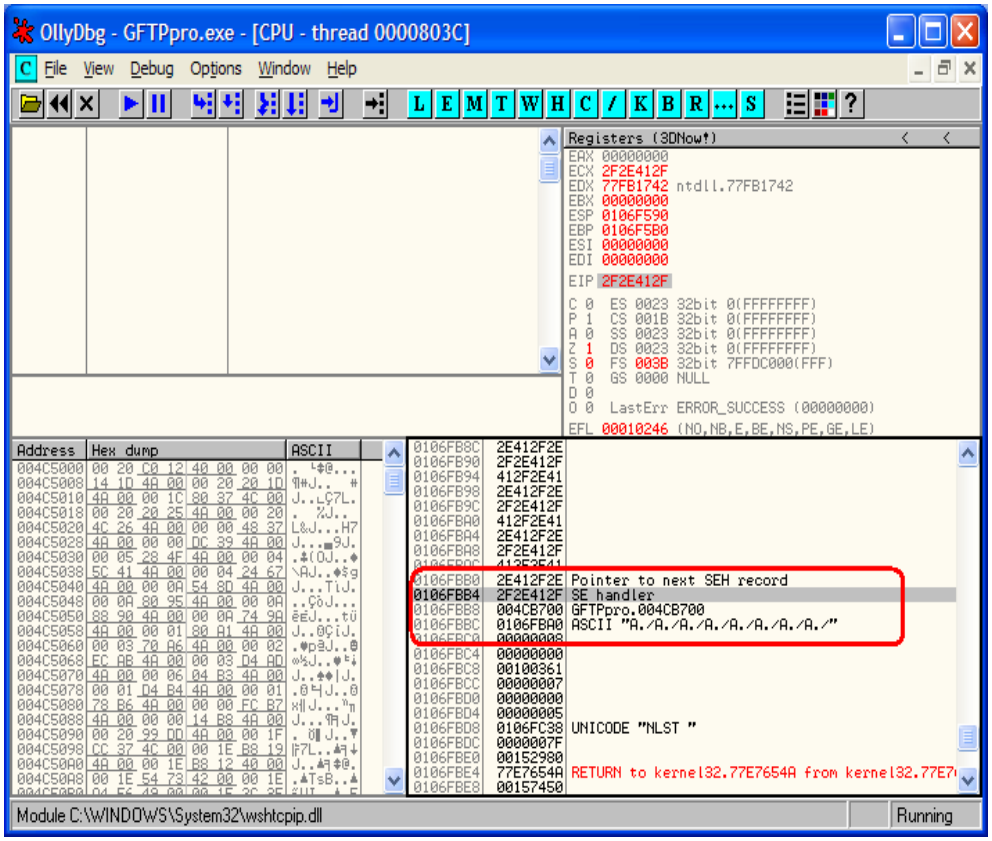
Kako je iz dobivenog ispisa vidljivo, poslužitelj prestaje odgovarati nakon označene linije odnosno NLST naredbe sa parametrom '**./A**' koji se ponavlja **330** puta. Analizom procesa

pomoću *OllyGdb debuggera* moguće je primijetiti da **EIP** registar sadrži vrijednost **0x0000412F** što označava ASCII vrijednost znakova '**/A**', odnosno testni unos koji je generirao *Fuzzer*. Pregledavanjem stoga očito je da je SEH (engl. *Structured Exception Handler*) prepisan sa vrijednostima **0x0000412F**. Stanje registara i izgled stoga prikazan je na slici (Slika 3).



Slika 3: SEH overflow

S obzirom da je konačni cilj potpuna kontrola **EIP** registra, potrebno je otvoriti konfiguraciju *Infigo FTPStress Fuzzera* i testnu veličinu **330** (pri kojoj se program srušio) promijeniti u **332**. Na taj način će *SEH handler* biti potpuno prepisan čime je u potpunosti moguće kontrolirati **EIP** registar. Potpuna kontrola **EIP** registra prikazana je na slici (Slika 4).

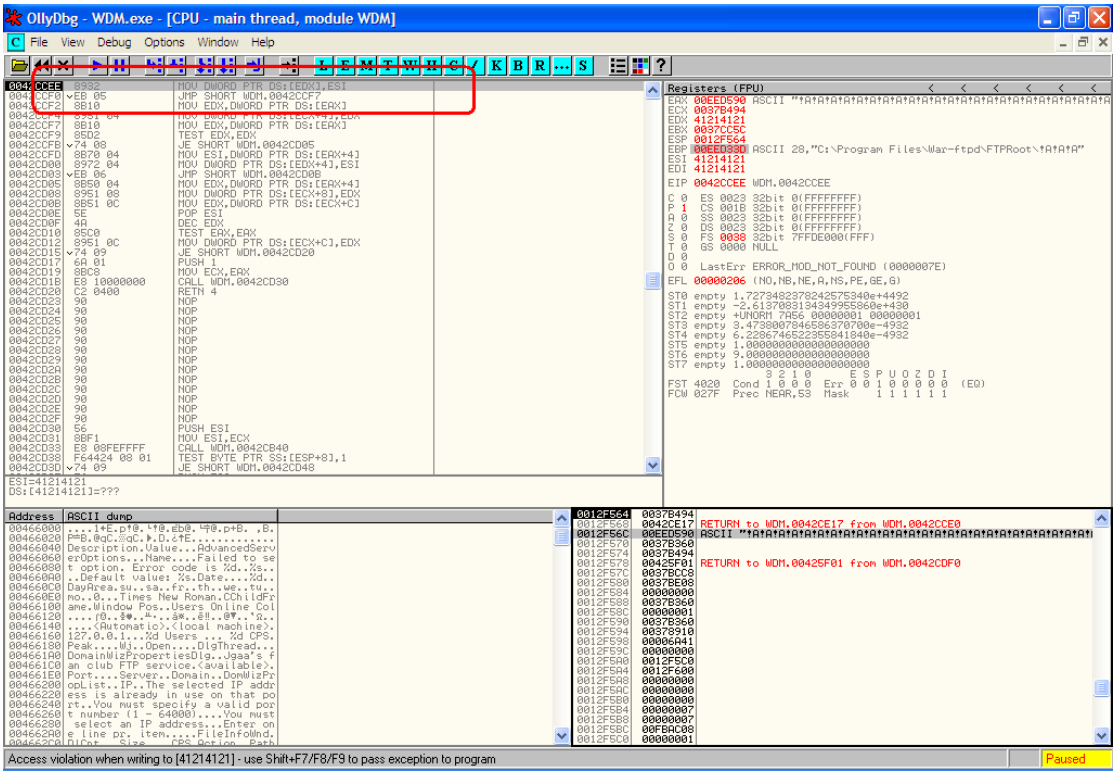


Slika 4: Potpuna kontrola EIP registra

S obzirom da su u ovom trenutku uglavnom poznate okolnosti koje uzrokuju preljev spremnika, ukoliko se želi nastaviti sa procesom testiranja Golden FTP poslužitelja za drugim ranjivostima, potrebno je samo isključiti '**./A**' testni podatak i '**NLST**' naredbu i nastaviti postupak *fuzzing*.

6.2. War FTP Daemon - WarFTPd

War FTP daemon program moguće je dohvatiti sa adrese http://www.warftp.org/. Iako je u prošlosti imao nekoliko kritičnih ranjivosti, u posljednje vrijeme nije bilo ozbiljnijih upozorenja o ranjivostima u ovom poslužitelju. Nakon određenog vremena primjene *fuzzing* metode na ovom poslužitelju, korištenjem *Infigo FTPStress Fuzzer* programa otkrivena je kritična ranjivost u *War Daemon Manageru*, alatu za održavanje *War Daemon FTP* poslužitelja. Nakon otprilike jedan sat *fuzzinga* FTP poslužitelja, **WDM.exe** proces je nasilno prekinuo s izvršavanjem. Analizom procesa lako je uočiti da se radi o preljevu spremnika. Na slici (Slika 5) može se vidjeti WDM proces u trenutku rušenja. Unos koji je generirao *Fuzzer* nalazi se u nekoliko registara. Između ostalih u **EDX** i **ESI** registrima. Obzirom da se program ruši pri izvršavanju instrukcije **MOV DWORD PTR [EDX], ESI,** moguće je prepisati 4 okteta bilo gdje u memoriji što je dovoljno za preuzimanje kontrole nad programom.



Slika 5: WDM.exe preljev spremnika

6.3. ARGOSOFT FTP SERVER

ArgoSoft FTP Server moguće je dohvatiti sa adrese http://www.argosoft.com. Ovaj FTP poslužitelj je prilično nezgodan za provođenje *fuzzinga*, jer ima mnogo sigurnosnih propusta pa se vrlo često ruši i teško ga je analizirati. Obzirom da *ArgoSoft FTP Server* ima problema u slučaju kada se na njega spoji *OllyDbg debugger*, nadgledanje ovog procesa provedeno je jednostavnijom inačicom alata za nadgledanje procesa pod nazivom *FaultMon* tvrtke EEye (http://www.eeye.com).

Nakon nekoliko minuta *fuzzinga*, proces nasilno prekida sa izvršavanjem i *FaultMon* prikazuje stanje procesora prilikom rušenja procesa. Pregledavanjem podataka kojima je *Infigo FTPStress Fuzzer* testirao FTP poslužitelj, lako je uočiti da je *ArgoSoft FTP Server* prekinuo s izvršavanjem nakon **RNTO** naredbe sa testnim podatkom '**&A**' veličine 3000 okteta.

```
C:\test\win\Chapter_14\bin>faultmon.exe -p 26844
…
pid=7E94 tid=83F8  EXCEPTION (unhandled)
-------------------------------------------------------------------
Exception C0000005 (ACCESS_VIOLATION writing [00000000])
-------------------------------------------------------------------
EAX=00000000: ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??
EBX=00000000: ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??
ECX=00260041: 01 90 01 00 01 00 01 00-01 A0 01 00 01 00 01 B0
EDX=77FB1742: 8B 4C 24 04 F7 41 04 06-00 00 00 B8 01 00 00 00
ESP=01111204: 2E 17 FB 77 E8 12 11 01-B0 F4 20 01 04 13 11 01
EBP=01111224: D0 12 11 01 00 17 FB 77-E8 12 11 01 B0 F4 20 01
ESI=00000000: ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??
EDI=00000000: ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??
EIP=00260047: 01 00 01 A0 01 00 01 00-01 B0 01 00 01 00 01 00
              --> ADD [EAX],EAX
'-------------------------------------------------------------------
```

Kao što je vidljivo iz primjera, **EIP** prilikom rušenja sadrži vrijednost **0x00260047** što je UNICODE vrijednost znakova '**&A**' + 6 što upućuje na testni podatak koji je generirao *Fuzzer*.

7. ZAKLJUČAK

Fuzzing tehnikom moguće je u vrlo kratkom vremenskom roku otkriti kritične sigurnosne ranjivosti u raznim programskim paketima kao što je i pokazano u ovom dokumentu. Važno je napomenuti da *fuzzing* tehnika, iako učinkovita, nije konačno rješenje za otkrivanje svih sigurnosnih propusta prisutnih u nekoj aplikaciji, te da je neke ranjivosti zbog njihove prirode nemoguće otkriti *fuzzingom*.

Fuzzing metoda prvenstveno se odlikuje svojom jednostavnošću i efikasnošću te mogućnosti automatizacije što znatno može ubrzati postupak otkrivanja sigurnosnih ranjivosti. Metoda je pogodna za velik broj različitih programskih rješenja i proizvoda, što potvrđuje i sve veći broj besplatnih i komercijalnih *fuzzer* programa koje je moguće pronaći na tržištu.