HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Analiza Format String sigurnosnog problema

CCERT-PUBDOC-2001-02-01

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.

Sadržaj

1. Uvod

Iskorištavanje sigurnosnog problema formata stringa predstavlja relativno novu tehniku koju neovlašteni korisnici mogu upotrebljavati na lokalnim ili udaljenim računalima. Ova tehnika javno je objašnjena prvi put u rujnu 1999. godine, no najviše pažnje privukla je javnim objavljivanjem exploita koji radi na wu-ftpd 2.6.0 poslužiteljima u lipnju 2000. godine. Ovaj exploit vrlo je vjerojatno napisan još puno ranije ali nije bio javno objavljivan do tada. Nakon toga, pojavio se veliki broj exploita zasnovanih na format string sigurnosnom problemu, kao što su:

- Napadi sa udaljenih računala:
  wu-ftpd, BSD ftpd, proftpd, rpc.statd, PHP 3 i 4, TIS-Firewall Toolkit
- Napadi sa lokalnog računala:
  lpr, LPRng, ypbind, BSD chpass i fstat, libc
- "ramen" crv također koristi format string sigurnosni problem koji postoji u wu-ftpd, rpc.statd i LPRng programskim paketima.

U ovom dokumentu objašnjena je ideja i analizirane su mogućnosti i ograničenja exploita baziranih na sigurnosnom problemu formata stringova. Na kraju je dan primjer napisanog exploita za implementaciju ftp poslužitelja Sveučilišta u Washingtonu – wu-ftpd-a inačice 2.6.0 koja dolazi standardno sa RedHat 6.2 distribucijom Linuxa.

2. Ideja exploita

U C programskom jeziku moguće je deklarirati funkcije koje imaju promjenjivi broj ulaznih parametara. Prilikom poziva te funkcije jedan stalni agrument mora reći funkciji koliko se argumenata nalazi u tom pozivu. Neke od poznatijih funkcija koje mogu imati promjenjivi broj ulaznih parametara su fprintf(), printf(), sprintf(), snprintf(), vprintf(), vsprintf(), vsnprintf(), setproctitle() i syslog(). Sve ove funkcije imaju dvije zajedničke karakteristike:

- Prvi parametar je tzv. format string.
- Konvertiraju sve argumente iz slijeda znakova nakon format stringa u izlazni slijed.

Zbog jednostavnosti u ovom će dokumentu biti prikazan primjer na printf() funkciji. Međutim, svi zaključci do kojih će se doći vrijede za bilo koju drugu funkciju formata koja je navedena.
Format string koristi se na dva načina:

- Ovaj podatak govori kako konvertirati argumente koji slijede u niz znakova (konverzija tipa, širine, preciznosti i drugo). Sintaksa za ovu konverziju identična je za sve gore navedene funkcije.
- Govori funkciji koliko zapravo argumenata slijedi nakon format stringa.

Sam format string je zapravo mješavina normalnih znakova koji se kopiraju u izlazni slijed kao i specifikatora konverzija koji se koriste kao znakovi za zamjenjivanje za slijedeće argumente. Navedeni program će ispisati brojeve 10 i 20 na stdout, odnosno ekran:

```
int i = 20;
int j = 10;
char *format_string = "Brojevi su %d i %d\n";
printf(format_string, i, j);
```

U ovom jednostavnom primjeru "%d" predstavlja specifikator konverzije. Specifikatori konverzije uvijek počinju sa znakom %. Znakovi koji slijede nakon % određuju stanja za izlazni slijed (širinu, preciznost i drugo) i specificiraju tip argumenta (int, float, char, char * i drugi). U izlaznom slijedu, svaki znak % zamjenjuje se sa vrijednošću odgovarajućeg argumenta (osim znakova %% koji označavaju % u izlaznom slijedu). Važni specifikatori konverzije su na primjer:

- %d – decimalni cijeli broj (int)
- %x – heksadecimalni cijeli broj (int)
- %s – string (char *)

Kao što je slučaj i sa sigurnosnim problemima prepisivanja podataka na stogu, i sa format string sigurnosnim problemima najveći je problem lošeg pisanja programa i ne provjeravanja ulaznih podataka. Primjer lošeg programa je:

```
char *korisnikov_ulaz;
[...]
printf(korisnikov_ulaz);
```

Ili još jedan primjer:

```
char *korisnikov_ulaz;
char *neki_string;
[...]
sprintf(neki_string, "%s", korisnikov_ulaz);
[...]
printf(neki_string);
```

U oba primjera korisnički ulaz koristi se kao format string argument prilikom poziva printf() funkcije. U ovim slučajevima trebalo bi primjeniti:

```
printf("%s", korisnikov_ulaz);
```

odnosno

```
printf("%s", neki_string);
```

za ispravnu upotrebu format stringa i izbjegavanje sigurnosnih problema vezanih za ove funkcije. Sada je potrebno vidjeti što se dešava sa navedenim primjerima kada korisnik unese %x znakove na ulazni string. U ovom slučaju printf() funkcija očekivati će neki cjelobrojni argument nakon format stringa što neće pronaći. Ovdje je također potrebno napomenuti da se ovakve greške nemogu pronaći za vrijeme prevođenja programa.
Primjer: wu-ftpd 2.6.0 ftp poslužiteljski program.
Problem u wu-ftpd 2.6.0 ftp poslužiteljskom programu nalazi se u vreply() funkciji (u src/ftpd.c datoteci). Pojednostavljena vreply() funkcija izgleda ovako:

```
void vreply([...], char *fmt, [...]);
{
        char buf[BUFSIZ];
        [...]
        snprintf(buf, sizeof(buf), fmt);
        [...]
```

U ovom slučaju site exec komanda *fmt sadrži niz znakova koje korisnik može upisati kao argument nakon "SITE EXEC" komande (sadržava je u site_exec() funkciji u src/ftpcmd.y datoteci). Na taj način ftp korisnik može kontrolirati format string koji se predaje u pozivu snprintf() funkciji.
U slijedećim točkama objašnjeno je kako neovlašteni korisnici mogu upotrebljavati ovaj sigurnosni problem na udaljenim računalima.

3. Čitanje stoga

Da bi se funkciji predali argumenti, dio koji poziva tu funkciju stavlja tzv. aktivacijski zapis (ili okvir) na stog. Npr, za funkciju f(int i, int *j) aktivacijski zapis za f() na stogu izgledao bi ovako:

Lokalne varijable, zapisani registri, aktivacijski zapisi drugih funkcija	Dno stoga
Pokazivač na j	
Vrijednost od i	
Povratna adresa na dio koji je pozvao f()	
Zapisani registri	
Lokalne varijable od f()	Vrh stoga

Ovdje je potrebno razjasniti što se dešava u slučaju da poziv printf() funkcije sadrži specifikatore konverzije bez odgovarajućih argumenata. Svi argumenti koji se predaju prilikom poziva printf() funkcije stavljaju se na stog. Funkcija printf() pretpostavlja da aktivacijski zapis sadrži argument na stogu za svaki specifikator konverzije u formatu stringa. Za svaki znak % čita se vrijednost na stogu sa određene lokacije. Na ovaj način printf() funkcija prolazi stog prema nižim adresama i čita odgovarajuće argumente sa stoga, te ih ispisuje u izlazni slijed znakova ne pregledavajući granice aktualnog aktivacijskog zapisa. U funkciji printf() nije implementirano provjeravanje granica.
U normalnim uvjetima rada format string sadrži informaciju o veličini aktualnog aktivacijskog zapisa kojeg je pozivajući dio programa stavio na stog. Manipulirajući format stringom neovlašteni korisnici mogu "uvjeriti" printf() funkciju da je njen aktivacijski zapis puno veći nego što je to u stvarnosti.
Na ovaj način neovlašteni korisnici mogu čitati vrijednosti sa stoga ukoliko se izlazni slijed printf() funkcije opet usmjeruje njima.
Primjer: wu-ftpd 2.6.0
Ovaj primjer pokazuje komunikaciju klijenta i ftp poslužitelja na Red Hat 6.2 Linux operacijskom sustavu. Umjesto pravog klijenta, korišten je program netcat. Unesene naredbe korisnika ispisane su podebljano. U ovom slučaju korisnik "bzdrnja" izdaje naredbu "SITE EXEC %x %x %x %x". %x znakovi interpretirani su kao format string i rezultat "31 bffff53c 1ee 6d" su zapravo vrijednosti sa stoga ftpd procesa.

```
% nc test 21
220 test FTP server (Version wu-2.6.0(2) Thu Aug 3 18:24:27 CEST
2000) ready.
USER bzdrnja
331 Password required for bzdrnja.
PASS test
230 User bzdrnja logged in.
SITE EXEC %x %x %x %x
200-31 bffff53c 1ee 6d
(end of '%x %x %x %x')
QUIT
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 291 bytes in 0 transfers.
221-Thanky you for useing the FTP service on test.
221 Goodbye.
```

3.1. Čitanje stringova sa skoro bilo koje lokacije u memoriji procesa

Ukoliko neovlašteni korisnik može vidjeti izlazni slijed funkcije printf(), on može doći i do više podataka od onih na stogu: znakovni nizovi sa manje ili više proizvoljnih lokacija u text ili data segmentu ili na heapu procesa također se mogu pročitati. Da bi analizirali kako ovo radi, presudno je razumjeti način na koji se argumenti znakovnih nizova predaju funkcijama.
Za znakovne nizove u argumentima aktivacijski zapis sadrži samo referencu (npr. pokazivač) na taj string. Dakle, da bi se znakovni string prikazao preko %s, u aktivacijski zapis mora biti upisan i

određeni pokazivač na taj string. Neovlašteni korisnici u ovom slučaju ne mogu promijeniti programski kod tako da se stavi dodatni pokazivač na stog prije poziva printf() funkcije. Jedina točka kontrole koju neovlašteni korisnici imaju u ovom je slučaju format string.
Format string je pohranjen na stogu. Precizno dodavajući %s specifikator sa dovoljno ostalih specifikatora (npr. %d ili %x) printf() se može natjerati na čitanje argumenata sa stoga sve do početka format stringa. Format string nalazi se u nekoliko okteta (4 na 32-bitnoj arhitekturi računala) koji zajedno čine pokazivač na memorijsku lokaciju koja sadrži znakovni niz koji je interesantan neovlaštenim korisnicima. Kada printf() počne interpretirati %s, čita točno ove oktete sa stoga uzimajući ih kao pokazivač na niz znakova.
Kao rezultat ovoga izlazni slijed printf() funkcije biti će:

Adresa stringa kopiranog kao znakovi na izlazni slijed.	Lokalne varijable, registri, povratne adrese koje se interpretiraju kao brojevi.	Stringovi koji su interesantni neovlaštenim korisnicima.

Ideja koja leži iza ovog trika je da se proširi aktivacijski zapis tako da sadrži barem početak format stringa. Na ovaj način neovlašteni korisnici mogu kontrolirati neke bitove aktivacijskog zapisa.

Format string	Dno stoga
Lokalne varijable, zapisani registri, aktivacijski zapisi drugih funkcija	
Pokazivač na format string	
Povratna adresa	
Registri i lokalne varijable printf() funkcije.	Vrh stoga

Kao što se primjećuje, ovu je tehniku moguće primijeniti samo ako je format string pohranjen na stog, tj. u lokalnu varijablu u funkciji. Osim toga, ako se izlazni slijed zapisuje u neki drugi spremnik (kao npr. sa sprintf() funkcijom), tada se može iskoristiti i taj spremnik.
Postoji još jedno ograničenje. Stringovi se u C programskom jeziku pohranjuju u ASCIIZ formatu. To znači da ovaj string ne može sadržavati niti jedan 0x00 oktet. Na 32-bitnim arhitekturama ovo znači da oko 2% sveukupnog adresnog prostora ne može biti pregledavano na ovaj način. Iako ovo možda izgleda nebitno, u praksi je pokazano da je veliki broj napada nemoguće izvesti na ovaj način budući da su u tim slučajevima interesantne informacije sadržane na niskim lokacijama).
Dodavanje velikog broja %x specifikatora prije %s da bi se pronašao format string na stogu može dovesti do stvaranja velikog format stringa. Ako je podržano od implementacije C standardne biblioteke (kao što je slučaj sa velikim brojem Linux distribucija), ovo može biti uvelike optimizirano korištenjem $ zastavice koja dozvoljava "skakanje" izravno na specificirani argument. Ovakva optimizacija ne samo da štedi prostor spremnika već i izbjegava nečitljiv izlazni slijed koji bi se inače sastojao od raznih lokalnih varijabli, registara i povratnih adresa koje bi se interpretirale kao cijeli brojevi.
Primjer: wu-ftpd 2.6.0
Ovaj put se koristi preciznije konstruirani format string kao argument za "SITE EXEC" komandu. AA dio se koristi samo za postavljanje precizno konstruiranog stringa. Stringovi "@eXX" i "pjXX" predstavljaju znakovni izgled pokazivača 0x08086a70 i 0x08086540 na pw_name i pw_passwd polja u passwd strukturi *pw. Oznaka "227$" koristi se umjesto 227 znakova %x. Kao što se može vidjeti "bzdrnja" je vrijednost pw->pw_name polja i "1IADJjdDOKDOw24445kkdOSDKKDSD" je vrijednost pw->pw_passwd polja (enkriptirana korisnička zaporka).

```
% nc test 21
220 test FTP server (Version wu-2.6.0(2) Thu Aug 3 18:28:27 CEST
2000) ready.
```

```
USER bzdrnja
331 Password required for bzdrnja.
PASS test
230 User bzdrnja logged in.
SITE EXEC AA@eXX%227$s
200-aaěXXbzdrnja
200 (end of 'aa@eXX%227$s')
SITE EXEC AapjXX%227$s
200-aapjXX$1$IADJjdDOKDOw24445kkdOSDKKDSD
(end of 'aapjXX%227$s')
QUIT
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 426 bytes in 0 transfers.
221-Thanky you for useing the FTP service on test.
221 Goodbye.
```

4. Pisanje cijelog broja na gotovo bilo koju memorijsku lokaciju procesa

Osim specifikatora konverzija koji su do sada spomenuti za jednostavne tipove znakovnih stringova postoji i jedan specifikator čija je funkcija drukčija: %n.
Primjer:

```
int i;
printf("12345%n", &i);
```

Kao što se vidi iz primjera, %n specifikator konverzije uzrokuje da se cijeli broj zapiše na lokaciju u memoriji. Budući da se format string nalazi negdje na stogu, neovlašteni korisnici mogu upotrebom ove tehnike kontrolirati pokazivač na cijeli broj. Ako se ispred %n znaka nalazi dovoljan broj %x znakova (ili $ oznaka) moguće je prolaziti vrijednosti stoga sve do točno određene lokacije gdje se nalazi format string. Format string počinje sa oktetima koji – interpretirani kao pokazivač – određuju memorijsku lokaciju na koju će se zapisivati.
Na ovaj način neovlašteni korisnici su u stanju:

- prepisati važne vrijednosti programa koje kontroliraju prava pristupa ili
- prepisati povratnu adresu na stogu, interne tablice linkanja (npr. ELF GOT ili PLT zapisi), pokazivače na funkcije ili setjmp/longjmp spremnike da bi izveli drukčiji tijek programa i skočili na eventualno dodani programski kod.

Vrijednost koja će se zapisati je određena sa brojem znakova upisanim prije %n specifikatora konverzije. Na ovaj način moguće je napisati proizvoljnu vrijednost na memorijsku lokaciju.
Prva mogućnost koja stoji na raspolaganju neovlaštenim korisnicima je korištenje nepotrebnih znakova. Da bi se zapisala vrijednost od recimo 1000, potrebno je na početak niza znakova koji se unose upisati 1000 nepotrebnih znakova. No, potrebno je obratiti pažnju i na broj %x specifikatora koji će se koristiti za prolaženje stogom i dohvaćanje format stringa. Zbog ovoga je %.8x dobro rješenje na 32-bitnim arhitekturama: njegova izlazna vrijednost je strogo 8 znakova i neovisno je od vrijednosti koja se upisuje; osim toga, može se koristiti i $ oznaka. Da bi se izbjegao dugačak format niza znakova, umjesto korištenja 1000 bespotrebnih znakova, može se koristiti specifikacija indikatora formata: po C standardu vrijednost koju zapisuje %n specifikator konverzije ovisi o broju znakova koji se moraju ispisati (C99 ISO standard) u izlazni slijed. Ako je stvarni izlazni slijed odrezan zbog granica izlaznog spremnika (npr. zbog vrijednosti "n" u snprintf() funkciji), to nema utjecaja na vrijednost koju zapisuje %n specifikator konverzije.
Dok je ovo u teoriji moguće provesti, u praksi je pokazano da ograničenja mnogih implementacija C standardne biblioteke ne mogu ispravno rukovati sa specifikatorima proizvoljnih dužina. U ovom slučaju potreban je drugi trik.
Drugi trik je da se %n specifikator koristi više od jednom: umjesto da se zapisuje samo jednom preko %n specifikatora, moguće je zapisati željeni string u nekoliko navrata gdje se pokazivači pomiču za po jednu adresu. Npr. na "little endian" 32-bitnoj arhitekturi to omogućava zapisivanje na nedozvoljene adrese (npr. zapisivanje na neparne adrese). Na ovaj način adrese se uvijek preklapaju za tri okteta,

ostavljajući jedan oktet ne diran za slijedeća zapisivanja. Između zapisivanja – tj. između različitih %n specifikatora, prvi trik se koristi za ispisivanje nepotrebnih znakova i za podešavanje vrijednosti okteta koji će biti ostavljen do slijedećeg zapisivanja – najmanje značajan oktet (LSB – eng. low significat byte) od sveukupnog broja okteta koji su zapisani. Na ovaj način za svaki oktet se može zapisati samo maksimum od 255 nepotrebnih znakova.
Za "big endian" arhitekturu zapisivanje se izvodi u drugom smjeru: svaki od pokazivača se umanjuje za jedan. Na 32-bitnim arhitekturama moguće je, dakle, izvesti samo dva zapisivanja gdje su pokazivači uvećani (ili smanjeni) za dva. Ovo, naravno, omogućava da se broj nevažnih znakova koje trebamo zapisati za ispunjenje spremnika može povećati do maksimuma od 65535 znakova.
Da bi se napravila četiri zapisa, format string mora sadržavati ASCII vrijednost četiri sukcesivna pokazivača. Niti jedan od njih ne smije sadržavati vrijednost 0x00, što u konačnici ostavlja oko 8% adresnog prostora u koji se ne može ništa zapisati ovom tehnikom (na 32-bitnoj arhitekturi računala).
Primjer: wu-ftpd 2.6.0
U ovom zadnjem primjeru korištena je opet "SITE EXEC" komanda koja je prepisala povratnu vrijednost na stogu. Nakon izvođenja site exec komande neovlašteni korisnik ima pristup ljuski sustava, unoseći komande "uname –a" i "id", u ovom slučaju kao administrator sustava.

```
% nc test 21
220 test FTP server (Version wu-2.6.0(2) Thu Aug 3 18:30:46 CEST
2000) ready.
USER bzdrnja
331 Password required for bzdrnja.
PASS test
230 User bzdrnja logged in.
SITE EXEC
AA@eXX%227$sDAKOFkfjgeorjwe9r9394jrjjasdfiaijwjOIRJWIRJJISDFIDFIjdfj
idjsfijsidjfisdfjsjosfijOIFSdJOIJFOIASJFoisdfjsdjdfiiidididi0000000ef
0e000000000000000000000000%^
200-
aaěXXbzdrnja0000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000asdoadoasofkkqwerw0e0qw0e00w0ew0eq00
we0q004305040%%%%#%&#%#%#%%%4$$$$$"#"#"#"#"##/bin/sh
200 (end of 'aa@eXX%227$s
dajsjsidijweijrjeijr$jierijjei%4$ji%Ijij5')
uname –a
Linux test 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
id
uid=0(root) gid=0(root) euid=500(bzdrnja) egid=500(bzdrnja)
groups=500(bzdrnja)
```