HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Analiza BOWall programskog paketa

CCERT-PUBDOC-2002-12-08

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

BOWall 1.2b je programski paket koji korisnicima Win NT i Win 2000 operativnih sustava pruža određeni nivo zaštite od napada prekoračenjem dimenzija spremnika (engl. buffer overflow).
Kako je poznato, sigurnosni propusti koji su posljedica neprovjerenih dimenzija spremnika, odnosno površnog programiranja, danas su najčešći problemi koji neovlaštenom korisniku omogućuju dolazak do ovlasti administratora sustava na udaljenom računalu.
Naime, propusti ovakvog tipa neovlaštenom korisniku omogućuju izvršavanje proizvoljnog koda na udaljenom računalu, što se najčešće koristi u svrhu ostvarivanja pristupa ljusci sustava sa ovlastima administratora.
Nakon što je to jednom postignuto napadač preuzima potpunu kontrolu nad cijelim sustavom, što mu dalje omogućuje spomenuto izvršavanje proizvoljnih komandi na ugroženom računalu.
Kako se može zaključiti, propusti ovog tipa vrlo su opasni, te stoga treba posebnu pažnju posvetiti njihovom uklanjanju. Klasičan način uklanjanja problema ovog tipa zasniva se na praćenju objavljenih sigurnosnih preporuka od strane proizvođača korištenih programskih paketa, te instalacijom adekvatnih sigurnosnih zakrpi koje uklanjaju objavljene probleme.
No, kako se može vidjeti kroz ovaj dokument postoje i programski alati koji korisnicima pružaju određeni nivo zaštite od napada ovakvog tipa, a jedan od njih je upravo BOWall 1.2b programski paket, koji je ujedno i tema ovog dokumenta.

2. Metode zaštite

Metode zaštite koje se koriste kod BOWall 1.2b programa baziraju se na dva osnovna pristupa:

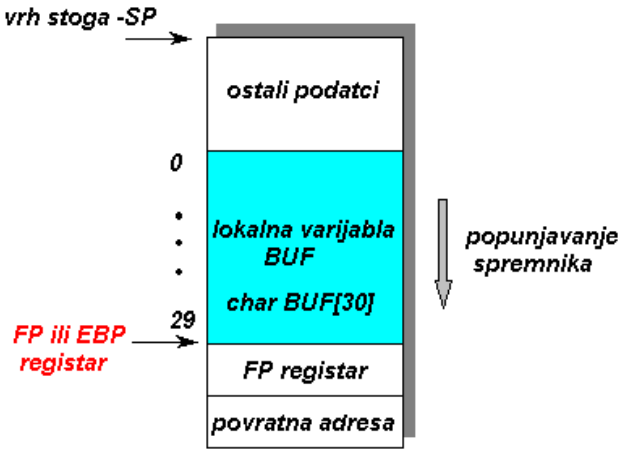
- praćenje i nadzor tijeka izvršavanja potencijalno opasnih programskih rutina, te njihova nadogradnja u svrhu uklanjanja detektiranih propusta,
- kontrolom poziva dinamičkih DLL funkcija, te njihova nadogradnja u svrhu uklanjanja uočenih propusta

2.1. Praćenje i nadzor tijeka izvršavanja programskih rutina

Ova metoda bazira se na praćenju tijeka izvršavanja programskih rutina za koje je poznato da su potencijalno opasne i osjetljive na napade ovog tipa.
Nakon što se program pokrene prati se izvođenje programskih rutina koje se nalaze u zadanom direktoriju, te se provjerava njihova osjetljivost na napad prekoračenjem dimenzija spremnika. Potencijalno opasne rutine prijavljuju se korisniku, nakon čega isti može pokrenuti proces uklanjanja uočenih problema.
Primjer takvih funkcija su programske rutine iz C jezika poput: strcpy, wstrcpy, strncpy, wstrncpy, strcat, wcscat, strncat, wstrncat, memcpy, memmove, sprintf, swprintf, scanf, wscanf, gets, getws, fgets, fgetws, te mnoge druge.
Najveći problem kod nabrojanih programskih funkcija je taj što niti jedna od njih kao argument ne prima dimenzije argumenata koji se u rutinu prenose.
Takav pristup neovlaštenom korisniku omogućuje da pažljivo osmišljenim napadom u takvu programsku rutinu proslijedi dovoljno velik niz podataka, koji će prepuniti spremnik rezerviran za njihovo lokalno pohranjivanje, što će dovesti do ranije opisanog problema.
Upravo se iz navedenog razloga u svrhu sigurnijeg programiranja preporučuje korištenje sličnih programskih rutina iste namjene, ali koje primaju dodatni argument, argument kojim se točno definira duljina niza koji se prenosi u funkciju (na primjer strncpy, strncat, snprintf programske rutine).
No, treba napomenuti da korištenje ovakvih 'sigurnijih' funkcija neće u potpunosti ukloniti probleme ovog tipa, budući da opet ovisi o samom programeru da li će biti ispravno implementirane sigurnosne provjere koje onemogućuju prekoračenje dimenzija spremnika, odnosno koje onemogućuju provođenje napada ovog tipa.
Ideja koja se koristi kod same metode je ta da se prati vrijednost elementa na stogu koji sadrži povratnu adresu prethodne rutine, te da se na temelju stanja toga polja spriječi daljnje izvršavanje programa ukoliko se uoče nepravilnosti za koje se smatra da mogu ugroziti sigurnost tijeka daljnjeg izvršavanja istog.

Naime, poznato je da se prilikom svakog poziva nove funkcije na stog pohranjuje povratna adresa slijedeće instrukcije koju je potrebno izvršiti nakon izlaska iz pozvane funkcije. To je potrebno kako bi se nakon izlaska iz pozvane programske rutine moglo dalje normalno nastaviti s izvođenjem programa.
Osim povratne adrese također se pri svakom pozivu nove funkcije pohranjuje i vrijednost FP (Frame Pointer) registra prethodne funkcije, registra koji omogućuje jednostavnije referenciranje lokalnih varijabli unutar trenutnog okvira na stogu.
Naime, korištenje ovog registra omogućuje jednostavnije referenciranje ostalih elemenata na stogu, poput lokalnih varijabli i povratnih vrijednosti, budući da su na ovaj način one uvijek jednako udaljene od FP registra, neovisno o ostalim operacijama koje se provode nad stogom.
Pohranjivanje ovog registra neophodno je kako bi se nakon izlaska iz funkcije moglo opet referencirati elementi stoga prethodne programske rutine.
Pogledajmo na primjeru kako izgleda stanje stoga prilikom poziva slijedeće programske rutine:

```
void func(void)
{
char buf[30];
gets(buf);
}
```



Slika 1: *Stanje na stogu za dani primjer*

Iz priložene slike može se uočiti da je nakon samog poziva funkcije prvo pohranjena vrijednost FP registra prethodne metode, a da nakon toga slijedi prostor koji je rezerviran za lokalne varijable pozvane funkcije. Također se može uočiti vrijednost povratne adrese na dnu slike, vrijednost koja je pohranjena netom prije poziva funkcije, odnosno izvršavanja call instrukcije.
Na samom vrhu stoga nalazit će se svi ostali podatci koji su rezultat izvođenja programskog koda same funkcije.
Ukoliko se pogleda programski kod u assembleru koji slijedi nakon poziva programske rutine, također se može uočiti upravo opisani slijed postupaka:

```
push ebp
mov ebp,esp
sub esp,30
```

Prva instrukcija pohranjuje vrijednost EBP (FP) registra prethodne funkcije, nakon čega slijedi postavljanje FP registra trenutne funkcije (koja se dobiva na temelju trenutnog stanja pokazivača vrha stoga), a na kraju se rezervira prostor za lokalne varijable ove rutine, što se postiže pomicanjem vrha stoga za 30 bajtova (koliko je velik lokalni spremnik buf u programskoj rutini func).

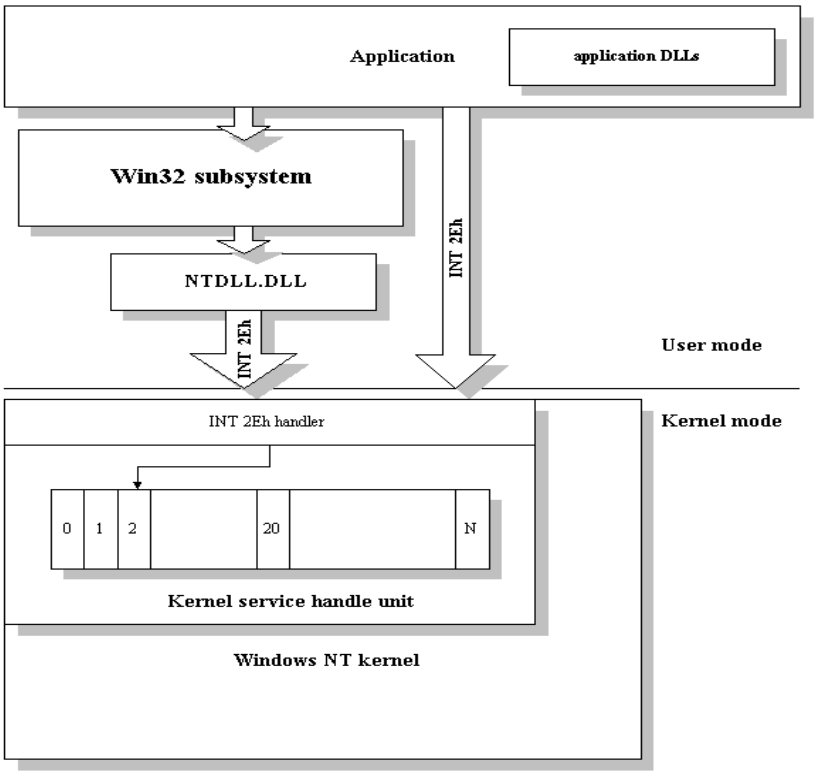
Ukoliko se sada pažljivo promotri struktura na slici može se uočiti da se pomoću FP registra mogu vrlo jednostavno referencirati sve lokalne varijable trenutne funkcije, bez obzira na operacije koje se provode nad stogom.
Ukoliko bi se u tu svrhu koristio SP registar, onda bi svaka nova operacija nad stogom mijenjala udaljenost (offset) od vrha stoga do traženih varijabli, što bi prilično zakompliciralo pristup drugim elementima stoga. Upravo zato je uvedeno korištenje FP registra.
Na ovaj način također se može uočiti da je povratna adresa od FP registra udaljena za 4 bajta, što omogućuje njezino jednostavno referenciranje u bilo kojem trenutku izvođenja programa.
Upravo tu činjenicu koristi ovaj program kako bi osigurao zaštitu od napada ovakvog tipa.
Postupak je slijedeći:

- pri pozivu funkcije zabilježiti vrijednost povratne vrijednosti
- izvršiti programski kod pozvane funkcije
- nakon izvršavanja koda ponovo dohvatiti vrijednost povratne adrese sa stoga
- usporediti ranije pohranjenu vrijednost sa trenutno dohvaćenom

Ukoliko se te vrijednosti razlikuju smatra se da je došlo do prepisivanje povratne adrese na stogu i izvršavanje programa se prekida, uz odgovarajuće upozorenje korisniku

2.2. Kontrola poziva dinamičkih DLL programskih rutina

Ova metoda bazira se na provođenju dodatnih sigurnosnih provjera prilikom poziva potencijalno opasnih DLL izvršnih datoteka iz dinamičkih biblioteka funkcija.
Naime, dodavanjem posebnog dijela koda takovim datotekama koji bi bio zadužen za provođenje sigurnosnih mjera pri pozivu potencijalno opasnih programskih rutina, omogućuje se pravovremena detekcija napada prekoračenjem dimenzija spremnika, odnosno pokušaj neregularnog korištenja stoga.
Treba napomenuti da implementacija ove metode zahtijeva nadogradnju postojećih DLL programskih rutina, sa dodatnim sigurnosnim dijelom koda koji bi obavljao potrebne sigurnosne provjere.
Metoda se bazira na pretpostavci da svaki ozbiljniji maliciozni program (kao i svaka druga aplikacija) za svoj rad mora koristiti dinamičke biblioteke funkcija.
Naime, poznato je kako kod Windows NT operativnih sustava postoje različiti sigurnosni modovi u kojima procesorska jedinica radi. Sve korisničke aplikacije rade na trećem sigurnosnom nivou sa relativno niskim ovlastima, dok sistemski pozivi samog operativnog sustava, te razni upravljački programi rade u nultom modu sa odgovarajućim višim ovlastima, budući da se radi o najnižim sistemskim resursima.
Samim time može se zaključiti kako je za bilo kakav pristup sistemskim resursima poput memorije i ostalih elemenata računalnog sustava, potrebno koristiti usluge samog operativnog sustava.
Na slici 2. dan je shematski prikaz u kojem se može uočiti upravo spomenuta hijerarhija izvršavanja.
Na priloženoj slici jasno se može uočiti kako se korisničke aplikacije izvršavaju u ne privilegiranom korisničkom modu, dok sama jezgra operativnog sustava sa pripadajućim modulima radi u privilegiranom sistemskom modu.
Također se može uočiti kako korisničke aplikacije putem Win32 podsustava (KERNEL.DLL) pristupaju NTDLL.DLL sučelju koje dalje omogućuje pristup najnižim sistemskim servisima.
Naravno, postoji i mogućnost direktnog pristupa sistemskim pozivima iz korisničkih aplikacija, no takav pristup koristi se samo u specifičnim situacijama.

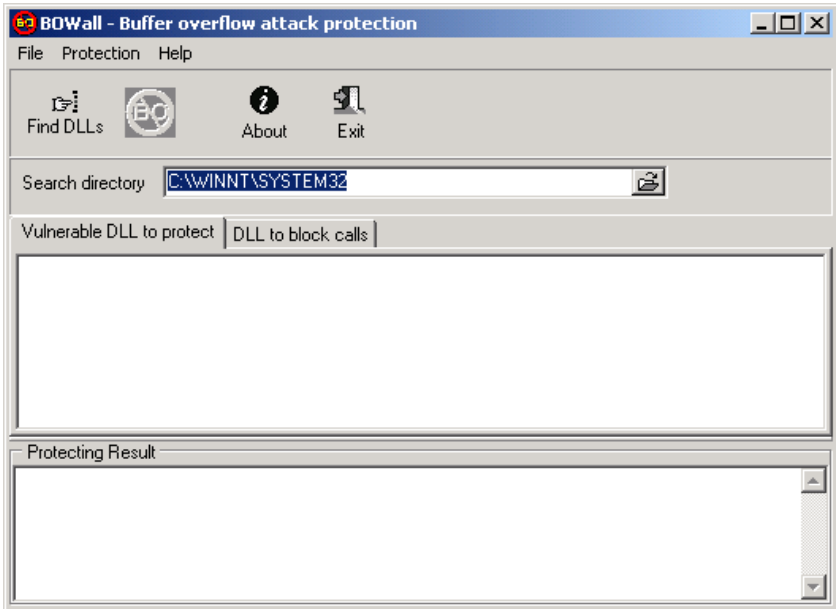


Slika 2: *Windows NT sustav*

Iz navedenog razmatranja može se zaključiti kako će svaka ozbiljnija maliciozna aplikacija za svoj rad morati koristiti usluge najnižih sistemski servisa, i upravo na toj ideji je baziran pristup koji koristi ovdje opisana metoda.
Naime, ideja je da se svaki dinamički modul nadogradi sa dijelom koda koji će provjeriti adresu programske rutine koja se poziva. Ukoliko se ta adresa nalazi negdje u podatkovnom dijelu programa koji se izvodi ili u području stoga smatra se da je došlo do prepisivanja stoga, te se prekida izvođenje programa.
Iako niti ova metoda nije savršena i sto posto pouzdana, u kombinaciji sa prethodnom metodom korisniku je na ovaj način pružen zavidan nivo zaštite u pogledu napada prekoračenjem dimenzija spremnika.

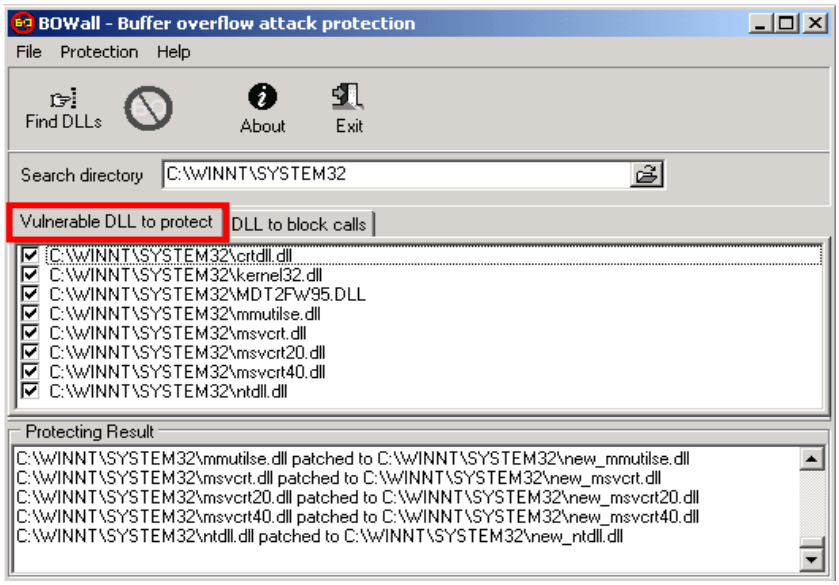
3. Opis programa

Nakon pokretanja BOWall 1.2b programa, otvara se klasično Windows grafičko sučelje (slika 3), koje korisniku pruža prilično jednostavno upravljanje ovim programom. Naime, program je sam po sebi vrlo jednostavan i intuitivan, te kao takav sadrži svega nekoliko opcija koje korisnik može odabrati prije početka postupka skeniranja.



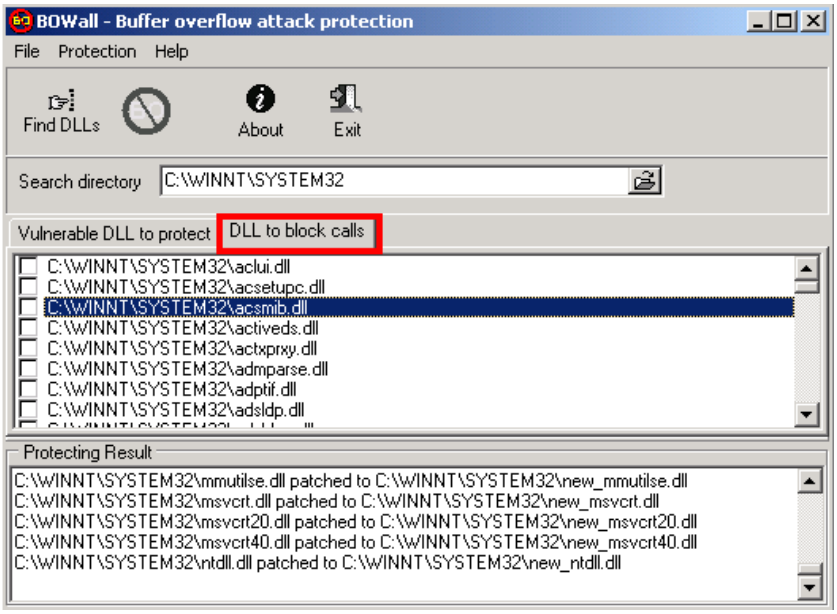
Slika 3: *Glavni prozor programa*

Unutar glavnog treba uočiti polje Search Directory, polje u koje se unosi ime radnog direktorija za ovaj program. Inicijalno ovo polje sadrži SYSTEM32 sistemski direktorij Windows operativnog sustava - C:\WINNT\SYSTEM32 .
Nakon što je definiran radni direktorij, potrebno je odabrati Find DLL's opciju pod karticom Protection, što će pokrenuti pregledavanje zadanog radnog direktorija.
Rezultati procesa pregledavanja, odnosno pretraživanja zadanog direktorija u svrhu pronalaženja potencijalno opasnih datoteka mogu se vidjeti u dva prozora.
Prvi prozor dobiva se pritiskom na karticu Vulnerable DLL to Protect (slika 4), i u njemu su prikazani rezultati testa bazirani na prvoj metodi (poglavlje 2.1), dok se drugi dobiva pritiskom na karticu DLL to Block Calls (slika 5), te prikazuje rezultate dobivene provođenjem metode opisane u poglavlju 2.2.



Slika 4: *Rezultati Vulnerable DLL to protect metode*

Nakon što su pronađene potencijalno opasne datoteke za obje primijenjene metode, korisniku su stavljene na raspolaganje opcije s kojima je moguće otkloniti detektirane probleme.



Slika 5: *Rezultati DLL to block calls metode*

Naime, u slučaju kada je aktivan prozor Vulnerable DLL to Protect (slika 4), moguće je odabrati opciju Protection -> Protect DLL's koja će popraviti potencijalno problematične datoteke.
Slično vrijedi i ukoliko je otvoren prozor DLL to Block Calls (slika 5), samo što se sada na istom mjestu nalazi opcija Protection -> Block DLL's.
Rezultati postupka popravljanja odabranih datoteka, mogu se pratiti u donjem okviru glavnog prozora, pod imenom Protecting Results (slika 6).
Također treba napomenuti da proces popravka potencijalno opasnih datoteka ne dira postojeće kopije uočenih DLL datoteka, već stvara njihove 'sigurnije' kopije. Ime novonastalih nadograđenih datoteka je:

```
new_<staro_ime_datoteke>
```

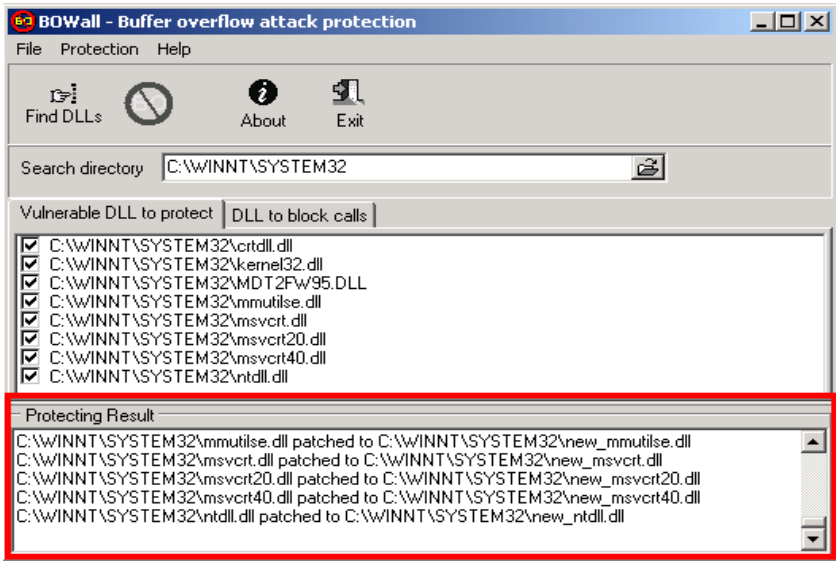
npr.

```
new_mmutilse.dll ili new_msvcrt.dll
```

Kako bi se napravljene promjene zaista aktivirale potrebno je obaviti slijedeće radnje:

- napraviti kopiju (engl. backup) originalnih datoteka, za koje se pokazalo da su osjetljive na napad prekoračenjem dimenzija spremnika
- preimenovati dograđene novonastale datoteke (new_<ime_datoteke>) u ime originalnih datoteka
- restartati računalo (engl. reboot)

Nakon toga aktivne bi trebale biti popravljene DLL datoteke, koje nisu osjetljive na napad prekoračenjem dimenzija spremnika. Zasad nisu još uočene nikakve nepravilnosti vezane za rad novo kreiranih DLL, koje bi mogle narušiti regularnost funkcioniranja samog sustava.



Slika 6: *Protecting Result okvir*

4. Zaključak

Na kraju se može zaključiti kako BOWall 1.2b programski paket predstavlja vrlo praktično rješenje za korisnike Win NT i Win 2000 operativnih sustava, u pogledu prevencije napada prekoračenjem dimenzija spremnika.

Iako metode koje program koristi u svrhu detektiranja potencijalno opasnih programskih modula nisu sto posto pouzdane, korištenje ovog programa u određenoj mjeri može zaštiti korisnike od napada prekoračenjem dimenzija spremnika.

U kombinaciji sa redovitim praćenjem objavljenih sigurnosnih zakrpi koje uklanjaju probleme ovog tipa, te njihova redovita instalacija, sigurnost se dodatno podiže na još viši nivo.

Također treba spomenuti vrlo jednostavno i intuitivno grafičko korisničko sučelje samog programa, što i manje iskusnim korisnicima olakšava njegovu upotrebu.