nAPI
Tehnička dokumentacija
Verzija <1.0>

Studentski tim: Davor Cihlar
Neven Ćubić
Berislav Hunjadi
Hrvoje Knežević
Ognjen Lajšić
Marko Plačko

Nastavnik: Leonardo Jelenković

Sadržaj

1.Opis razvijenog proizvoda

Programska potpora nAPI oblikovana je za praćenje rada 32-bitnih urođenih izvršnih datoteka pod operacijskim sustavom Windows. Moguće je pratiti Windows API funkcije vezane uz datoteke, memoriju, dretve i priključnice (engl. sockets). Za svaku funkciju nAPI može generirati različite opcije. Korisnik odlučuje koje će opcije nadzirati. Nakon pokretanja izvršne datoteke preko nAPI-ja, generira se statistika za one opcije koje je korisnik izabrao. Takvu funkcionalnost nAPI može postići modifikacijom tablice adresa vanjskih funkcija (engl. Import Address Table) koju sadrže izvršne datoteke. Modifikacija je moguća samo kod onih datoteka koje su urođene, tj. generirane baš za IA32 arhitekturu. Jezgra programske potpore promijeni adrese funkcija tako da pokazuju na vlastite promijenjene funkcije, a ne više na originalne Windows API funkcije. Vlastite funkcije pozivaju originalne Windows API funkcije, ali pritom dobivaju sve parametre koje im proslijedi nadziran program. Jezgra pritom može računati različite statistike koje se kasnije i ispisuju u zapisnik. Jezgra je komandnolinijski program, a grafičko sučelje je napravljeno s ciljem lakše interakcije s korisnikom.

Podržane su sljedeće funkcije:

1. datoteke: CreateFile, ReadFile, WriteFile, CloseHandleFile, DeleteFile, MoveFile, CopyFile, CreateDirectory i RemoveDirectory (podržane su i ANSI i UNICODE verzije funkcija)
2. memorija: GlobalAlloc, GlobalFree
3. dretve: GetModuleHandle, CreateThread, CloseHandle, ExitThread
4. priključnice: socket, bind, listen, accept, connect, send, sendto, recv, recvfrom, closesocket

2.Tehničke značajke

Dio programske potpore koji korisnik vidi je grafičko sučelje. Da bi funkcionalnost bila moguća, potrebno je generirati konfiguraciju koju je zadao korisnik označivši određene opcije u GUI-ju. Ta konfiguracija je ulazni parametar za jezgru programske potpore na temelju koje se mijenja IAT i kasnije ispisuje statistiku. Datoteka sa statistikom je ulazni parametar za GUI pri ispisu konfiguracije. Na taj način je omogućen rad nAPI-ja.

Program "starter" pokreće zadani proces u suspendiranom načinu i nakon toga injektira svoj kod za učitavanje jezgre (dll-a "nAPI.dll"). Kod se ubacuje tako da se rezervira mjesto u udaljenom procesu dovoljno veliko da u njega stane injektirani izvršni kod. Osim toga, potrebno je i alocirati memoriju za sam naziv dll-a jer udaljeni proces ne može bez posebnih funkcija pristupati memoriji drugog udaljenog procesa. Nakon alokacije zapisuje se kod i podaci u tu memoriju.

Da bi proces mogao ispravno nastaviti s radom, ali uz nadzor, potrebno je prebaciti EIP registar udaljenog procesa na početak injektiranog koda, te inicijalizirati stog tog udaljenog procesa.

Asemblerski kod radi sljedeće: dohvati sa stoga adresu funkcije LoadLibraryA, te ju pozove. Taj API zatim sa stoga dohvati adresu stringa "nAPI.dll" i učita ga. Zatim taj naš asemblerski program dohvati staru vrijednost izmijenjenih registara, te napravi return (koji sa stoga dohvati adresu koja je baš početak originalne aplikacije).

Nakon što je taj kod injektiran i svi podaci inicijalizirani, možemo reći procesu da nastavi s izvršavanjem, te ga sačekamo da izvrši.

Jezgra (nAPI.dll) pri učitavanju još izvrši svoju inicijalizaciju u kojoj učita nAPI.conf, inicijalizira log, te prepravi IAT. Prepravlja se samo IAT nadziranog procesa, ne i IAT-i dll-a koji su još učitani uz taj proces, te je stoga nadziranje ponekad ograničeno.

IAT tablica sadrži adrese svih API funkcija koje proces koristi. Svaka nadzirana funkcija se traži u IAT, te se zamijeni sa adresom zamjenske funkcije.

Zamjenske funkcije imaju istu funkcionalnost kao i originalne Windows API funkcije. Osim toga, one prikupljaju podatke i rade statistike. Provjeravaju uspješnost izvršavanja funkcije i ispisuju opis greške u slučaju da je funkcija vratila grešku. Podaci koji se skupljaju su npr. ukupan broj primljenih ili poslanih byteova, trajanje izvođenja funkcije, broj otvorenih priključnica, datoteka itd. Ti podaci se zapisuju u zapisnik (klasa koja prikuplja podatke i sprema ih) te se kasnije zapisuju u datoteku izvještaja.

U svrhu praćenja određenih statistika koriste se globalne varijable. Za datotečne funkcije to su statistički podaci o pročitanim i zapisanim podacima, za memorijske funkcije to su statistike o veličini i broju alociranih blokova, odnosno prosječnoj veličini alociranih blokova, dok su za socket funkcije to podaci o broju otvorenih socketa, broju poslanih i primljenih podataka te broju slanja i primanja.

Sve te globalne varijable su zaštićene kritičnim odsječkom da bi se mogle nadzirati i višedretvene aplikacije. Osim globalnih varijabla, zasebnim kritičnim odsječkom je i zaštićeno zapisivanje u zapisnik kako se ne bi izmiješali podaci.

Sve funkcije koje se mogu mijenjati i njihove pripadne zamjenske funkcije su nabrojane u datoteci ReplacementList.cpp. Zamjenske funkcije su podijeljene u različita polja ovisno o tome iz kojeg dll-a je funkcija koju mijenjaju. Osnovna podjela je na funkcije iz kernel32.dll koje su sadržane u polju kernel32_functions[], funkcije iz user32.dll koje su sadržane u polju user32_functions[] i funkcije iz ws2_32.dll i wsock32.dll koje su sadržane u ws2_32_functions[] polju. Na primjer funkcija nAPI_WriteFile mijenja funkciju WriteFile koja se nalazi u kernel32.dll pa je ona navedena u polju podataka kernel32_functions[]. Tamo su sadržane sve zamjenske funkcije koje se odnose na kernel32.dll zajedno sa imenima originalnih funkcija koje mijenjaju. Ideja je da se naziv funkcije koja se želi pratiti zada u konfiguracijskim parametrima programu. Program će pomoću navedenih struktura na temelju imena funkcije znati točno koju zamjensku funkciju mora pozvati. Na primjer ako mu je zadan parametar WriteFile on će u polju kernel32_functions[] naći element koji odgovara tom imenu i uzeti naziv zamjenske funkcije koju mora pozvati. Smještaj pojedinih funkcija ovisi o tome na što se odnose.

Datotečne funkcije su smještene u datoteci FileAPIOverride.cpp, memorijske u datoteci MemoryAPIOverride.cpp, dretvene funkcije u ThreadAPIOverride.cpp, a funkcije koje se odnose na rad sa priključnicama ("socketima") se nalaze u SocketAPIOverride. Funkcija WriteFile se nalazi u datoteci FileApiOverride.cpp pa će program tamo tražiti zamjensku funkciju za nju. Kada program koji se prati pri radu pozove funkciju WriteFile pozvat će se funkcija nAPI_WriteFile koja se nalazi u toj datoteci. Ta funkcija će pozvati originalnu funkciju WriteFile iz kernel32.dll i stvoriti će novi unos u izvještaju koji će ovisno o konfiguracijskim parametrima programa sadržavati ime datoteke u koju se piše, broj poziva funkcije, količina zapisanih podataka i slične statističke podatke. Parametri koji se mogu pratiti ovise o funkciji koju se prati i svi su navedeni u dijelu 2.1. Na navedenom primjeru vidljivo je da je funkcionalnost originalnih funkcija potpuno sadržana s obzirom da one same izvode zadani posao. Jedina promjena u programu koja se radi nad programom koji se prati je u IAT tablici. Funkcija nAPI_WriteFile će pozvati originalnu funkciju i pritom vratiti u pozivajući program rezultat te funkcije, dok će sama WriteFile funkcija obaviti zadani posao.

U programu je izvedena podrška za datotečne funkcije ANSI i Unicode formata, budući da većina nadziranih datotečnih funkcija postoji u oba oblika. Postoje zamjenske funkcije za oba formata. Ako se poziva ANSI funkcija, onda se njezini parametri i rezultat izvođenja samo proslijede u zamjensku funkciju. Ako se pozove Unicode funkcija, konvertiraju se svi stringovi u običan ANSI format i potom se pozove zamjenska funkcija, kojoj se prosljeđuju svi parametri originalne funkcije, kao i rezultat njezina izvođenja.

Prilikom poziva CreateFile funkcije, dakle prilikom otvaranja nove datoteke ili otvaranja postojeće, handle na datoteku i ime datoteke pohranjuju se u listu handleova. Implementirane su dvije takve liste, lista datotečnih handleova i lista dretvenih handleova, u koju se pohranjuju handleovi na dretvu i pripadajući kontekst dretve prilikom poziva funkcije CreateThread. Prilikom zatvaranja datoteke ili dretve poziva se ista API funkcija CloseHandle. Kako bi se odredilo radi li se o handleu za dretvu ili za datoteku, vrši se provjera prema zapisima u listama, te ukoliko zapis bude pronađen, handle se briše iz liste.

2.1Testiranje

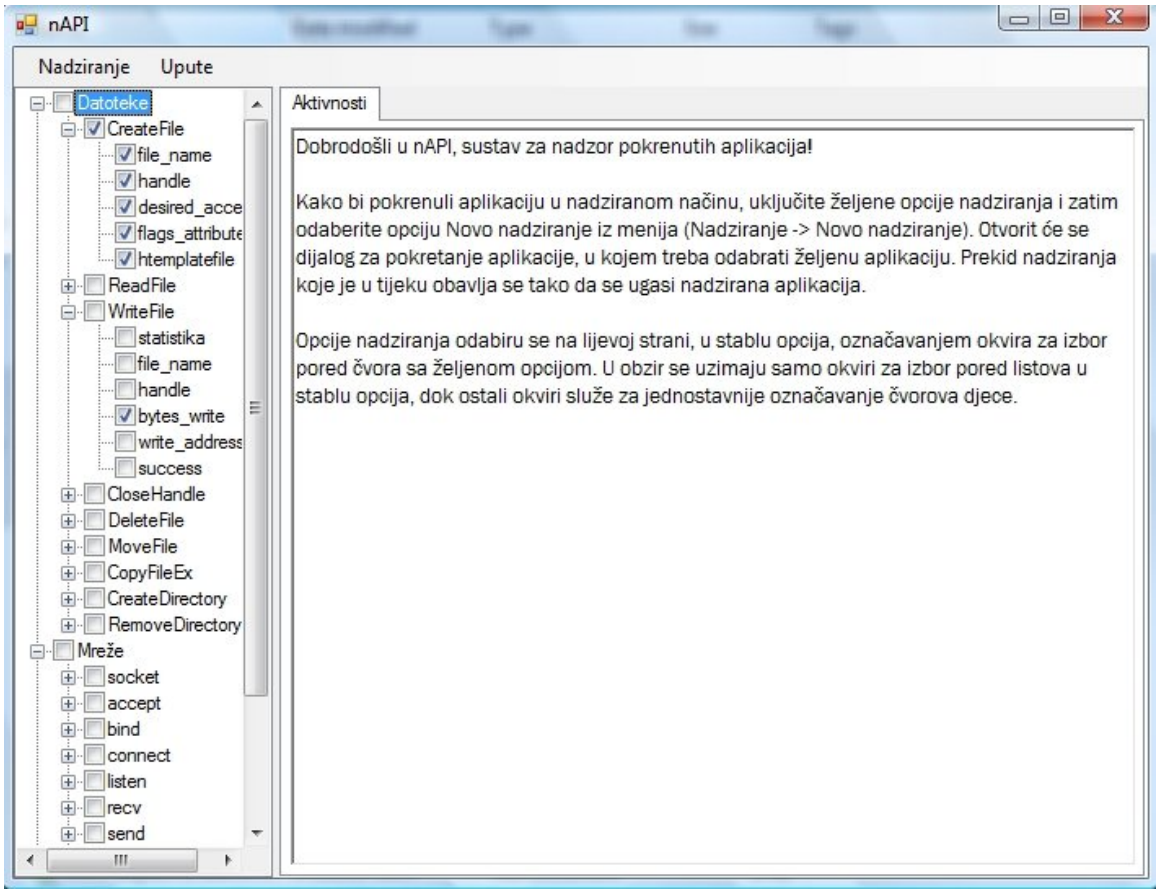
Prije izdavanja završne verzije programa provedena su testiranja da bi se dokazalo da program radi svoj posao ispravno. Posebno su provedena testiranja za datotečne, dretvene, memorijske i funkcije za priključnice.

Testiranja za datotečne funkcije su se izvodila pomoću većeg broja manjih programa koji testiraju po nekoliko funkcija iz skupa datotečnih funkcija koje nAPI prati. Prilikom testiranja datotečnih API-a programi za testiranje su rađeni na način da su davali ispis parametara sa kojima se pozivaju originalne funkcije. Cilj je bio posebno ispisati parametre koji se zadaju funkciji prije nego što se pozove. Na taj način kad nAPI pozove zamjensku funkciju i unese te parametre u zapisnik, može se napraviti usporedba sa ispisom na konzoli u cilju provjeravanja ispravnosti rada nAPI-a. To pokazuje da li je pozvana zamjenska funkcija primila ispravne parametre, ispravno ih unijela u zapisnik i konačno da li je sa ispravnim parametrima pozvala originalnu funkciju. Uz provjeru zapisa u zapisnik dodatno testni programi pokušavaju ispitivati ponašanje nAPI-a kad pozvana funkcija vrati grešku, te ispravnost unosa u zapisnik kad se dogodi pogreška. Kako je cilj testova ispitati ponašanje nAPI-a kad funkcija ne obavi svoj posao i vrati grešku, a ne testirati rad originalnih API funkcija, dovoljno je bilo testirati bar jedan slučaj pogreške po zamjenskoj funkciji i pogledati unos u zapisnik i ponašanje nAPI-a. Kao primjer testirano je ponašanje kada se prati funkcija CreateFile koja pokuša otvoriti već otvorenu datoteku koja ima namještene atribute koji to ne dopuštaju. Funkcije ReadFile i WriteFile mogu nailaziti na slučajeve kad pokušaju čitati ili pisati u dokument koji nema dopušteno čitanje ili pisanje što rezultira pogreškom, te pritom je bitno provjeriti je li nAPI to ispravno detektirao provjeravajući ukupnu količinu zapisanih, odnosno pročitanih podataka iz datoteke. Funkcija DeleteFile može vratiti grešku ako pokušava izbrisati otvorenu datoteku, a funkcija CreateDirectory ako pokušava napraviti već postojeći direktorij. Program je prošao sva prva testiranja, ispravno bilježi podatke u zapisnik i ne ometa rad originalnih API funkcija.

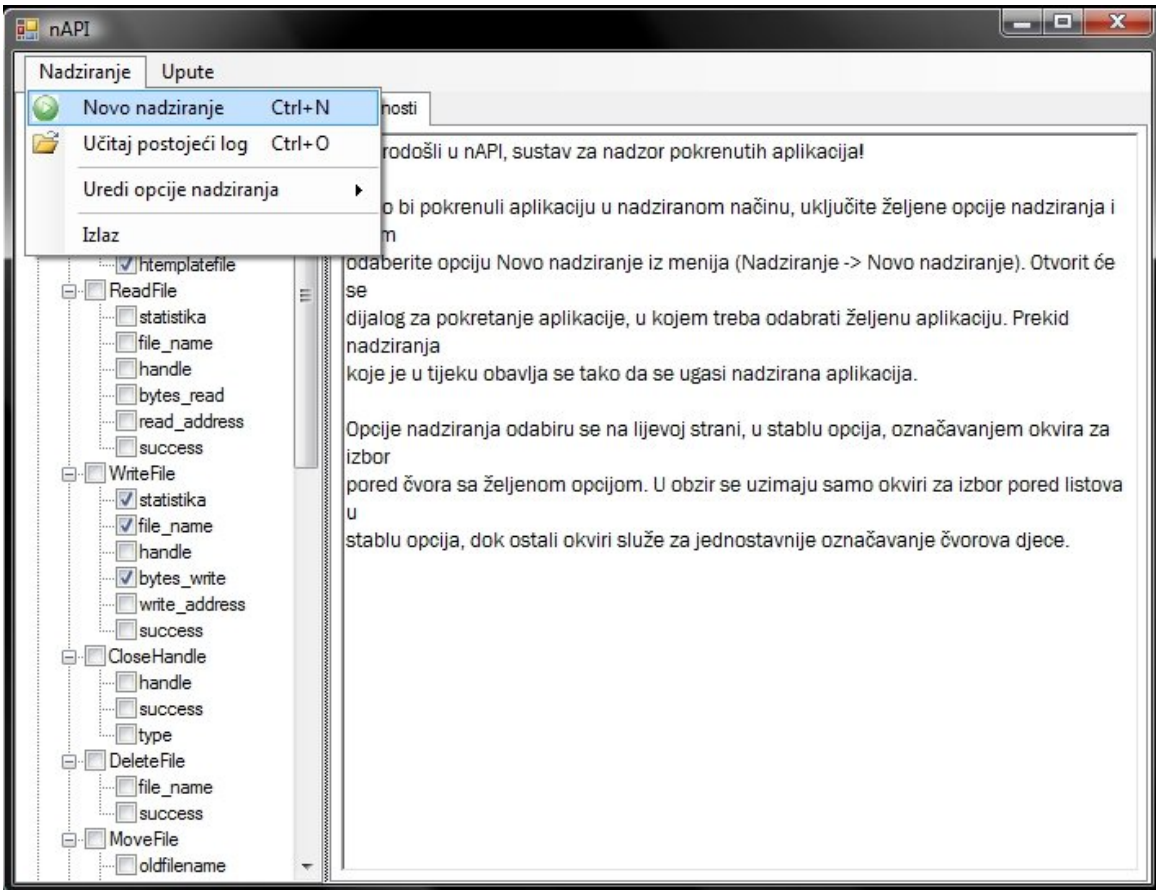
Testiranja za priključnice su se vršila na sličan način. Testni programi su ciljano trebali izazvati grešku u nadzornom programu, izvoditi ilegalne operacije sa priključnicama, ...

3.Upute za korištenje

Pri pokretanju GUI aplikacije, prikazat će se ekran dobrodošlice.

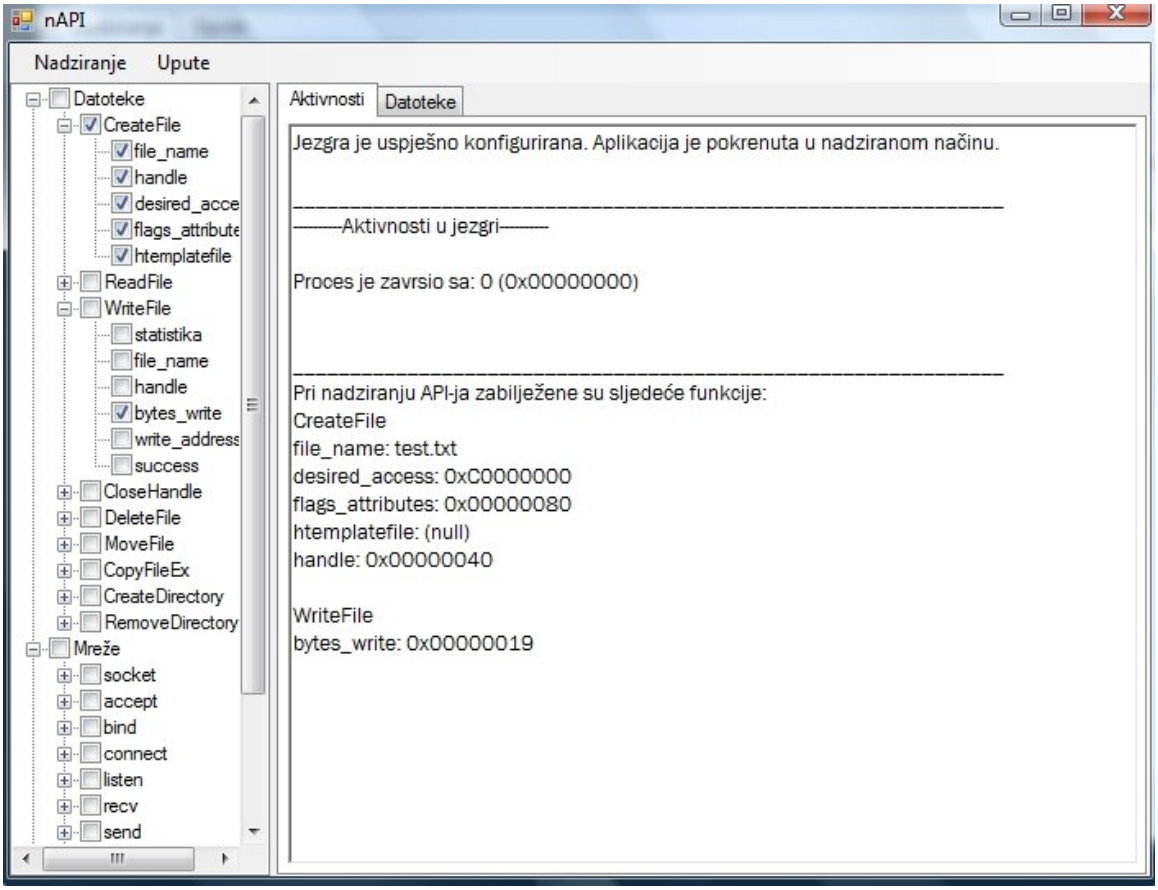


Kako bi pokrenuli aplikaciju u nadziranom načinu, uključite željene opcije nadziranja i zatim odaberite opciju Novo nadziranje iz menija (Nadziranje → Novo nadziranje).



Otvoriti će se dijalog za pokretanje aplikacije, u kojem treba odabrati željenu aplikaciju. Prekid nadziranja koje je u tijeku obavlja se tako da se ugasi nadzirana aplikacija.

Po završetku rada nadzirane aplikacije završava nadziranje i prikazuje se ispis zapisnika u desnom dijelu sučelja. Osim toga se prikažu i nove kartice zavisno od toga što je odabrano za praćenje.



Opcije nadziranja odabiru se na lijevoj strani, u stablu opcija, označavanjem okvira za izbor pored čvora sa željenom opcijom. U obzir se uzimaju samo okviri za izbor pored listova u stablu opcija, dok ostali okviri služe za jednostavnije označavanje čvorova djece.

Ostale opcije koje su ponuđene u izborniku *Nadziranje* su učitavanje postojećeg zapisnika u aplikaciju, te izbornik za uređivanje opcija nadziranja, pomoću kojeg vrlo jednostavno možemo označiti sve funkcije određene vrste i sve njihove parametre u stablu opcija.

3.1Pokretanje iz konzole

Aplikaciju se može nadzirati i iz konzole. Potrebno je pozicionirati se u mapu u kojoj se nalazi izvršna datoteka aplikacije za nadzor, te pokrenuti Starter.exe, i to tako da mu se kao ulazni parametar preda lokacija programa na disku koji se želi nadzirati, te iza nje argumenti koje želimo proslijediti nadziranom programu. Npr.

```
Starter "c:\mapa_X\naziv_pracenog_programa.exe" argument1 argument2 ...
```

Po završetku praćenja, bit će stvoren zapisnik oblika:

```
nAPI-"naziv_praćenog_programa.exe".txt
```

u istoj mapi u kojoj se nalazi Starter.exe.

Da bi program ispravno radio, potrebno je prije pokretanja ispuniti konfiguracijsku datoteku nAPI.conf u formatu koji će biti kasnije objašnjen.

3.2Formati datoteka

Program nAPI se konfigurira pomoću datoteke nAPI.conf. Prije nego počne s radom, nAPI iz te datoteke učita podatke koji mu govore koju funkciju mora pratiti i koje parametre za tu funkciju mora upisivati u zapisnik. Format datoteke je sljedeći:

Prvo mora biti napisan broj parametra koji se zadaje nAPI-u da prati, nakon toga se unosi ime funkcije koja se mijenja i parametri koji se upisuju u zapisnik. Odjednom se može zadati više funkcija koje se prate, pri čemu svaka funkcija mora biti napisana u novom redu sa svim svojim podacima. Primjer ispravno zadane nAPI.conf datoteke:

```
1 CreateFileA handle
3 MoveFileA oldfilename newfilename Success
3 CopyFileExA originalfilename newfilename Success
2 DeleteFileA file_name success
1 CloseHandle handle
```

Korisnik ne mora ručno mijenjati nAPI.conf datoteku, to se izvodi automatski preko grafičkog sučelja kad se odaberu funkcije i parametri koji se prate.

U konfiguracijskoj datoteci podržana je i opcija `/all/`, koja u log datoteku upisuje sve parametre za nadziranu funkciju. Korisnik ne mora ručno mijenjati nAPI.conf datoteku, to se izvodi automatski preko grafičkog sučelja kad se odaberu funkcije i parametri koji se prate.

Zapisnik se generira nakon završetka izvođenja praćenog programa. On sadrži zapise poziva funkcija koje se prate po redu pozivanja prilikom izvođenja programa. Uz njih ispisuje i podatke o parametrima koji su navedeni u konfiguracijskoj datoteci za svaku funkciju. Naziv zapisnika ovisi o nazivu programa koji se prati i sljedećeg je formata:

```
nAPI-"naziv_praćenog_programa.exe".txt
```

Primjer zapisnika nakon završetka izvođenja pračenog programa:

```
1 CreateFileA
handle: 0x000000B8

11 WriteFile
handle: 0x000000B8
file_name: " 1
test.txt
"
bytes_write: 0x0000001F
write_address: 0x0012FEFC
Success: 1
sum_read: 0
num_read: 0
avg_read: 0
sum_write: 31
num_write: 1
avg_written: 31

11 ReadFile
handle: 0x000000B8
file_name: " 1
test.txt
"
```

```
bytes_read: 0x0000000B
read_address: 0x0012FEF8
Success: 1
sum_read: 11
num_read: 1
avg_read: 11
sum_write: 31
num_write: 1
avg_written: 31
```

3.3Funkcije koje su podržane i njihovi parametri

---Datoteke---

CreateFile

file_name - ime datoteke koju kreiramo

handle - ručica na datoteku

desired_access - razina pristupa - za čitanje, pisanje ili oboje

flags_attributes - zastavice i atributi datoteke

htemplatefile - handle na read-only template datoteku

ReadFile

statistika - ukupno i prosječno pročitano (u bajtovima) te broj poziva fje

file_name - ime datoteke iz koje čitamo

handle - handle na datoteku

bytes_read - broj pročitanih bajtova

read_address - adresa s koje se čita

success - uspješnost čitanja (1 ako je datoteka uspješno pročitana, 0 ako nije)

WriteFile

statistika - ukupno i prosječno zapisano (u bajtovima) te broj poziva fje

file_name - ime datoteke u koju pišemo

handle - handle na datoteku

bytes_write - broj zapisanih bajtova

write_address - adresa u memoriji na koju pišemo

success - uspješnost pisanja (1 ako je u datoteku uspješno pisano, 0 ako nije)

CloseHandle

handle - handle na datoteku koju zatvaramo

success - uspješnost zatvaranja (1 ako je handle uspješno zatvoren, 0 ako)

type - tip handlea (handle na datoteku ili na dretvu)

DeleteFile

file_name - ime datoteke koju brišemo

success - uspješnost brisanja (0 ako se radi o pokušaju brisanja otvorene ili nepostojeće datoteke, inače 1)

MoveFile

oldfilename - puni file path datoteke koju premještamo

newfilename - file path na koji premještamo datoteku (datoteka može biti premještena pod novim imenom)

success - uspješnost premještanja (1 ako je datoteka uspješno premještena, 0 ako datoteka ne postoji ili je pokušavamo premjestiti na nepostojeći path)

CopyFileEx

originalfilename - puni file path datoteke koju kopiramo

newfilename - file path na koji kopiramo datoteku (datoteka može biti kopirana pod novim imenom)

flags - zastavice koje određuju neke parametre pri kopiranju

success - uspješnost kopiranja (1 ako je datoteka uspješno kopirana, 0 ako datoteka ne postoji ili je pokušavamo kopirati na nepostojeći path)

CreateDirectory

path_name - path direktorija koji kreiramo

attributes - atributi direktorija (ako je NULL onda se direktoriju pridodjeljuju defaultni atributi)

success - uspješnost kreiranja (1 ako je uspješno kreiran, 0 ako direktorij već postoji ili ako kreiramo direktorij na nepostojećem pathu)

RemoveDirectory

path_name - path direktorija kojeg brišemo

success - uspješnost brisanja (1 ako je uspješno obrisan, 0 ako direktorij ne postoji ili ako postoji barem jedan otvoreni handle na navedeni direktorij)

---Mreže---

---

socket

socket - opisnik socketa

success - uspješnost obavljanja funkcije

family - familija adresa (IPv4, IPv6)

type - UDP ili TCP

protocol - protokol koji se koristi

brojsocketa - broj socketa za vrijeme programa

accept

success - uspješnost obavljanja funkcije

newsocket - opisnik socketa na kojem se izvodi komunikacija

listensocket - socket na kojem je osluškivano

adresa - IP adresa računala koje se spojilo na server

port - port računala koje se spojilo na server

addrlen - duljina strukture u kojoj su adresa i port

brojsocketa - broj socketa za vrijeme programa

bind

success - uspješnost obavljanja funkcije

socket - opisnik socketa

adresa - adresa na kojoj se vrši povezivanje

port - port na kojem se vrši povezivanje

namelen - duljina strukture u kojoj su adresa i port

connect

success - uspješnost obavljanja funkcije

socket - opisnik socketa

adresa - adresa na koju se računalo spojilo

port - port računala na kojeg smo spojeni

namelen - duljina strukture u kojoj su adresa i port

listen

success - uspješnost obavljanja funkcije

socket - opisnik socketa

maxconnect - maksimalna duljina reda čekanja konekcija

recv

statistika - broj poziva funkcije recv, prosjek primljenih podataka po pozivu funkcije u bajtovima i ukupan broj primljenih bajtova

vrprimanja - vrijeme i datum kad su primljeni podaci

success - uspješnost obavljanja funkcije

brzina - brzina prijenosa podataka (bajtovi u sekundi)

vrijeme - vrijeme potrebno za prijenos podataka (mikrosekunde)

socket - opisnik socketa

flags - način primanja

primljeno - broj primljenih bajtova

recvptr - kazaljka na početak niza koji je primljen

send

statistika - broj poziva funkcije send, prosjek poslanih podataka po pozivu funkcije u bajtovima i ukupan broj poslanih bajtova

vrslanja - vrijeme i datum kad su poslani podaci

success - uspješnost obavljanja funkcije

brzina - brzina prijenosa podataka (bajtovi u sekundi)

vrijeme - vrijeme potrebno za prijenos podataka (mikrosekunde)

socket - opisnik socketa

flags - način primanja

poslano - broj poslanih bajtova

sendptr - kazaljka na početak niza koji je poslan

recvfrom

statistika - broj poziva funkcije recvfrom, prosjek primljenih podataka po pozivu funkcije u bajtovima i ukupan broj primljenih bajtova

vrprimanja - vrijeme i datum kad su primljeni podaci

success - uspješnost obavljanja funkcije

brzina - brzina prijenosa podataka (bajtovi u sekundi)

vrijeme - vrijeme potrebno za prijenos podataka (mikrosekunde)

socket - opisnik socketa

recvptr - kazaljka na početak niza koji je primljen

primljeno - broj primljenih bajtova

flags - način primanja

adresa - IP adresa računala od kojeg su primljeni podaci

port - port sa kojeg se šalju podaci

addrlen - duljina strukture u kojoj su adresa i port

sendto

statistika - broj poziva funkcije sendto, prosjek poslanih podataka po pozivu funkcije u bajtovima i ukupan broj poslanih bajtova

vrslanja - vrijeme i datum kad su poslani podaci

success - uspješnost obavljanja funkcije

brzina - brzina prijenosa podataka (bajtovi u sekundi)

vrijeme - vrijeme potrebno za prijenos podataka (mikrosekunde)

socket - opisnik socketa

sendptr - kazaljka na početak niza koji je poslan

poslano - broj poslanih bajtova

flags - način primanja

adresa - IP adresa racunala na koje se šalju podaci

port - port na koji se šalju podaci

addrlen - duljina strukture u kojoj su adresa i port

closesocket

socket - opisnik socketa

success - uspješnost obavljanja funkcije

ukvrijeme - ukupno vrijeme od socket() do closesocket() (mikrosekunde)

avgbrsl - srednja brzina slanja podataka (bajtovi u sekundi)

avgbrpr - srednja brzina primanja podataka (bajtovi u sekundi)

---Memorija---

GlobalAlloc

statistika - broj alociranih blokova, ukupno i prosječno alocirano bajtova

flags - postavke za tip bloka memorije

size - veličina alociranog bloka memorije

handle - vraćeni pokazivač

GlobalFree

statistika - broj oslobođenih blokova, ukupno i prosječno oslobođeno bajtova

handle - handle na blok koji se oslobađa

result - uspješnost oslobađanja (null ako je uspješno)

found - broj oslobođenih bajtova ako se oslobodio alocirani blok (0 ako taj blok nije bio dobiven s GlobalAlloc)

---Dretve---

GetModuleHandle

name - ime učitanog modula (dll-a) za kojeg se želi doznati handle (null znači da se traži handle od trenutnog exe-a)

handle - dobiveni handle (lokacija modula u memoriji)

CreateThread

statistika - ukupan broj dretvi

stack - količina memorije zahtjevane za dretvu (0 = default stack size)

function - pointer funkcije koja predstavlja novu dretvu

flags - postavke za tip bloka memorije

ID - Unique ID dretve

current_thread - ID dretve u izvođenju

CloseHandle

statistika - ukupan broj dretvi

exit_code - vrijednost s kojom je dretva završila (nije uvijek definirano)

type - THREAD_HANDLE

ID - ID dretve koja je završila

ok - vrijednost koja govori je li se dretva uspješno izvršila (tj. je li uopće bila stvorena pomoću CreateThread)

handle - handle dretve koja se zatvara

current_thread - ID dretve u izvođenju

ExitThread

statistika - ukupan broj dretvi

exit_code - vrijednost s kojom je dretva završila (nije uvijek definirano)

current_thread - ID dretve u izvođenju

4.Literatura

Johannes Passing, Using Import Address Table hooking for testing, http://jpassing.wordpress.com/2008/01/06/using-import-address-table-hooking-for-testing/, 1.6.2008.

Razni slični projekti se mogu naći na internetskoj stranici: http://www.mvps.org/vbvision/Sample_Projects.htm