HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Anti-IDS sustavi

CCERT-PUBDOC-2001-11-07

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

Anti IDS (engl. *Intrusion Detection Systems*) sustavi, kako i sam naziv kaže, su sustavi čiji je osnovni cilj zaobilaženje sigurnosnih mjera koje provode sustavi za detekciju neovlaštenog mrežnog prometa.
Ideja koja se primjenjuje kod svih takovih sustava je u osnovi ista. Mrežni zahtjevi pokušavaju se što više maskirati, kako bi im se na taj način omogućilo uspješno zaobilaženje sigurnosnih provjera raznih IDS sustava, a da opet na drugu stranu budu dovoljno regularni, te da će biti prepoznati i obrađeni bez greške od strane mrežnih poslužitelja.
Unutar ovoga dokumenta detaljno su opisani pristupi koje koriste anti IDS sustavi u svrhu zaobilaženja detekcije neovlaštenog web prometa. To znači da će biti detaljno opisani postojeći mehanizmi, te konkretne implementacije ovakvih sustava, s time što je u ovom slučaju cilj isključivo web servis, odnosno HTTP protokol.
Kao primjer uzeta je konkretna implementacija jednog takvog anti IDS sustava pod imenom whiskers, sustava koji omogućuje zaobilaženje sigurnosnih mjera koje provode IDS sustavi, ukoliko se radi o web prometu, odnosno HTTP protokolu.
Iako je u ovom slučaju naglasak dan isključivo na zaobilaženju detekcije od strane IDS sustava, isti pristup se može koristiti i u slučaju mrežnih filtra i analizatora (*sniffers*), zatim analizatora logova, te ostalih sigurnosnih sustava koji interpretiraju sadržaj web prometa, u svrhu podizanja sigurnosti na viši nivo kod ovog vrlo popularnog mrežnog servisa.
Također treba napomenuti da se slični postupci analize, te metode i teorija iznesena u ovome dokumentu, uz manje modifikacije može primijeniti i na druge mrežne protokole, iako je ovaj dokument vezan isključivo za web servis, i njemu pripadajući HTTP protokol.

2. Osnovni pojmovi

U svrhu lakšeg praćenja iznesenog materijala, bitno je poznavati komponente jednog tipičnog HTTP zahtjeva. Na slijedećem primjeru biti će u tu svrhu ukratko pojašnjeni elementi jednog tipičnog HTTP zahtjeva:

```
GET /cgi-bin/ime_skripte.cgi HTTP/1.0
```

- na samom početku upita nalazi se *metoda* zahtjeva. Tipične vrijednosti ovog elementa su `GET`, `PUT`, `POST` i sl., a moguće su i druge vrijednosti definirane HTTP protokolom. Na temelju na ovaj način navedene metode, web poslužitelj zna interpretirati daljnje elemente istog zahtjeva,
- nakon metode slijedi URI adresa (engl. Uniform Resource Identificator) web resursa. URI, ili kako se vrlo često koristi kao sinonim URL (Uniform Resource Locator) adresa, je znakovni niz kojime se identificiraju razni tipovi web resursa (dokumenti, slike, servisi, skripte, i sl.). Svaka URI adresa može biti ili relativna (npr. http://ime_poslužitelja/nesto/ime_dokumenta), ili apsolutna (npr. /nesto/ime_dokumenta). Iako se često URL i URI adrese koriste u istom kontekstu, URL se zapravo odnosi na apsolutni URI, u kojem je uključena i točna specifikacija servisa na koji se adresa odnosi (npr. ftp, http, ...),
- zadnji element navedenog zahtjeva predstavlja korištenu *verziju* HTTP protokola, a sintaksa je uvijek oblika `HTTP/x.x`. Moguće vrijednosti `x.x` polja su `0.9`, `1.0`, `1.1`. Ukoliko je navedena verzija `0.9`, upit se često naziva jednostavnim HTTP upitom,
- svi ovi gore izneseni elementi zajedno formiraju jedan HTTP zahtjev, sa naglaskom da su elementi odvojeni razmakom.

Kroz ovaj dokument spominjat će se dva osnovna tipa IDS sustava: surovi (engl. *raw*) i pametni (engl. *smart*), te će ovdje u kratko biti iznesene njihove osnovne karakteristike.

2.1. Smart IDS (Intrusion Detection Systems)

Osnovna karakteristika smart IDS sustava, je ta da oni u potpunosti razumiju, a samim time i interpretiraju određeni protokol (u našem slučaju HTTP).

Ova grupa IDS sustava temeljito analizira svaki mrežni zahtjev koji pripada dotičnom protokolu, te na temelju tako provedenih analiza donosi odluke vezane za regularnost primljenog paketa.
Odluke se donose na temelju usporedbe analiziranih paketa i lokalno pohranjenih potpisa (engl. signature), koji sadrže tipične karakteristike mrežnog prometa vezanog za taj protokol.
To bi ujedno značilo da se smart IDS sustavi ponašaju identično kao i obični poslužitelji za dotični mrežni servis, ali sa nešto manjom brzinom rada, odnosno manjom količinom procesiranih mrežnih zahtjeva u jedinici vremena.
Nešto manja brzina rada biti će posljedica dodatnog programskog koda, generiranog u svrhu sigurnosnih provjera koje se provode za svaki mrežni zahtjev.
Primjeri IDS sustava koji spadaju u ovu skupinu su Dragon i Snort sustavi za detekciju neovlaštenog mrežnog prometa.
Raw IDS sustavi mogu se jednostavno zamisliti kao neka vrsta filtra mrežnog prometa, budući da oni jednostavno skeniraju pristigle pakete, te u njima traže neke ključne znakovne nizove, na temelju kojih će se donijeti odluka o prihvaćanju ili odbijanju istog paketa.
Samim time ovi su sustavi mnogo brži od ranije spomenutih smart sustava, budući da se ne troši vrijeme na interpretiranje sadržaja samog protokola, već se čisto analizira bajt po bajt svakog pristiglog paketa.
To ujedno podrazumijeva da kod ovakvih sustava ne postoji nikakva ugrađena inteligencija, već se njihov rad bazira na čistom surovom procesiranju mrežnih paketa.
Primjer IDS sustava koji spada u ovu skupinu je RealSecure IDS sustav.
Obje od gore navedenih skupina IDS sustava imaju svoje prednosti i nedostatke, te se mogu prevariti na različite načine. Kako je već ranije spomenuto, anti IDS sustavi u svrhu zavaravanja koriste razne mehanizme maskiranja mrežnih zahtjeva, kako na taj način ne bi odgovarali niti jednom potpisu na temelju kojeg bi IDS sustav uočio maliciozne karakteristike procesiranog paketa.
Možda bi trebalo pojasniti već nekoliko puta korišten pojam potpisa. Naime, potpisi se mogu shvatiti kao obični znakovni nizovi koje IDS sustav koristi kao pokazatelj malicioznog mrežnog prometa, te ih adekvatno tome pokušava locirati unutar primljenog zahtjeva, kako bi na temelju toga upozorio na neželjeni mrežni promet.
Ukoliko se uoči takav paket koji unutar sebe sadrži poznati neregularni znakovni niz, zahtjev se tretira kao pokušaj neovlaštenog pristupa poslužitelju, te se odbija i prijavljuje kao pokušaj neovlaštenog pristupa zaštićenim mrežnim resursima.
Također treba napomenuti da vrijedi jedno općenito pravilo. Što su znakovni nizovi koji se pretražuju duži, manja je vjerojatnost da će isti pojaviti unutar mrežnog zahtjeva, te da će ga IDS sustav kao takvog detektirati.
Upravo se zato većina IDS uređaja konfigurira na način da pokušavaju locirati vrlo kratke znakovne nizove unutar primljenog paketa, kako bi se na taj način uočila što veća količina malicioznog prometa.
Primjer takovih kratkih znakovnih nizova (potpisa) su čista imena direktorija, gdje se onda unutar primljenog paketa traži znakovni niz, koji bi indicirao da se radi o pokušaju neovlaštenog pristupa zaštićenom direktoriju (npr. /phf direktorij).
S druge strane takav pristup će vrlo često rezultirati odbijanjem posve regularnih mrežnih paketa, što ponekad može bitno utjecati na performanse poslužitelja.
Na primjer, gore navedeni potpis kojim se želi detektirati neovlašteni pristup /phf direktoriju zadovoljavati će između ostalih i dolje navedeni HTTP zahtjev, te će adekvatno tome biti prijavljen i eventualno odbačen, što možda nije ono što uistinu želimo učiniti sa ovim paketom.

```
GET /phfiles/phonefiles.txt HTTP/1.0
```

Upravo zato proizvođači IDS sustava trebaju posebnu pažnju posvetiti konfiguraciji njihovih sustava, pogotovo kada se ide na ovakvo skraćivanje potpisa, budući da to vrlo često može imati neželjene posljedice.
Vrlo često proizvođači IDS sustava uzimaju /cgi-bin/ kao jedan od potpisa, što može biti problem ukoliko se CGI skripte ne nalaze unutar navedenog /cgi-bin/ direktorija.
Kada se govori o postojećim implementacijama IDS sustava vrlo je teško govoriti o samim detaljima implementacije, budući da iz razumljivih razloga programski kod nije dostupan.

3. Principi rada

3.1. Detekcija metode HTTP zahtjeva

Prilikom testiranja ovakvih sigurnosnih sustava, za većinu njih se pokazalo da sadrže propust, koji je posljedica činjenice da se pretražuju samo zahtjevi sa GET HTTP metodom, što se nikako ne može tretirati kao dobro rješenje. Naime, većina sustava uzimala je zahtjeve oblika:

```
GET /cgi-bin/some.cgi
```

kao one, koji mogu potencijalno maliciozni.

Ovakav pristup se pokazao kao lošim rješenjem, budući da su zahtjevi koji koriste neku drugu metodu (npr. HEAD ili POST) , a ne GET, prolazili nedetektirani, odnosno smatrali su se regularnima, što nije ono što se željelo postići.

Na primjer, isti zahtjev ali sada u nešto modificiranom obliku, prošao bi bez ikakvih problema kod većine IDS sustava.

```
HEAD /cgi-bin/some.cgi
```

Upravo ovakav pristup se koristio unutar whisker anti IDS sustava, kako bi se testirale nepravilnosti i uočili propusti u implementacijama raznih IDS sustava. Naime, pravilno formirani potpisi na temelju kojih se odlučuje o prihvaćanju, odnosno odbijanju paketa nikako ne bi smjeli uključivati HTTP metodu kao dio potpisa.

No, moglo bi se s druge strane reći da to i nije toliki problem, budući da je za iskorištavanje sigurnosnih propusta kod CGI sučelja potrebno koristiti GET metodu.

Na žalost, to nije potpuno točna tvrdnja budući da postoje mehanizmi kojima se može ugroziti CGI sučelje i putem POST odnosno HEAD HTTP zahtjeva, ovisno o kodiranju same CGI skripte.

3.2. URL kodiranje

Poznato je da URI specifikacija (RFC 1738) sadrži određene nedostatke s obzirom na činjenicu da dozvoljava korištenje samo određene skupine alfanumeričkih znakova, dok HTML standard podržava kompletni ISO-8859-1 skup znakova.

Tako se unutar URL adrese mogu naći samo znakovi iz skupine [0-9, a-z, A-Z] alfanumeričkih znakova, te neki dodatni specijalni znakovi (?, $, !, + ...), što odudara od HTML specifikacije.

Upravo se u tu svrhu koristi URL kodiranje, kako bi se na taj način omogućilo korištenje i ostalih znakova unutar URL adrese. U ovakvom pristupu svaki kodirani znak sastoji se od znaka %, nakon kojeg slijedi dvije heksadecimalne znamenke, koje opisuju taj znak prema ISO Latin skupu znakova (npr %xx).

Whisker alat koristi upravo URL kodiranje kao pomoćni alat u svrhu zavaravanja IDS sustava. Naime, ova metoda zavaravanja sastoji se u tome što se osim znakova koji zahtijevaju ovakav način kodiranja, kodiraju i oni posve regularni, koji su inače potpuno podržani od strane URI specifikacije.

Tako na primjer raw IDS sustavi, koji pretražuju cgi-bin potpis unutar primljenog paketa neće uopće detektirati isti niz, koji bi bio URL kodiran, odnosno u obliku "%63%67%69%2d%62%69%6e".

Za razliku od raw IDS sustava, smart sustavi bi detektirali ovakav pokušaj prijevare, budući da se ponašaju identično kao i pravi web poslužitelji, te kao takvi provode URL dekodiranje, što bi rezultiralo detektiranjem kodiranog potpisa.

No, u posljednje vrijeme su propusti ovog tipa uklonjeni kod gotovo svih iole kvalitetnijih IDS sustava, te se upravo opisan pokušaj zavaravanja IDS sustava više ne koristi.

Usprkos tome whisker alat implementira opisanu metodu, navođenjem –l opcije.

3.3. Dvostruki / znak

Ovaj pristup zaobilaženja detekcije od strane IDS sustava, bazira se na zamijeni svih slash (/)znakova unutar HTTP upita, sa dvostrukim slash znakom (//) unutar istog upita.

Ovakav pristup rezultirat će činjenicom da će se sva pretraživanja /cgi-bin/ime_skripte.cgi potpisa pokazati neuspješnima ukoliko se formira nešto modificirani oblik istog upita, npr. //cgi-bin//ime_skripte.cgi.

Većina današnjih implementacija IDS sustava (raw i smart) uspješno detektiraju ovakve pokušaje neovlaštenog pristupa, te ostale derivacije ove metode koje koriste trostruke, ili druge višestruke slash znakove.
Raw sustavi u ovom kontekstu često emuliraju smart sustave, ili jednostavno administratoru prijavljuju postojanje višestrukih slash znakova unutar upita, što predstavlja jasno upozorenje da se može raditi o pokušaju neovlaštenog pristupa web poslužitelju.
Whisker ne implementira direktno ovu metodu, budući da su implementirane dodatne metode koje se baziraju na sličnom pristupu, a mogu dati kvalitetnije krajnje rezultate.

3.4. Proširivanje HTTP zahtjeva

Sljedeća tehnika koja se upotrebljava u svrhu zavaravanja IDS sustava, je ta da se originalni upit nadopuni dodatnim proizvoljnim znakovima, koji će onemogućiti ispravnu detekciju malicioznog HTTP zahtjeva. Na primjer zahtjev oblika

```
GET /cgi-bin/blahblah/../some.cgi HTTP/1.0
```

prošao bi sasvim neopaženo, a nakon reverse traversal procesiranja na strani poslužitelja, rezultirao bi posve valjanim upitom za /cgi-bin/some.cgi skriptom.
Ovakav pokušaj prijevare uočiti će svaki smart IDS sustav (kako i ime samo ime govori), dok će raw sustavi primijetiti neregularnosti u upitu i prijaviti greške poput postojanja /.../ niza u primljenom HTTP upitu, što predstavlja pokazatelj da se radi o eventualnom pokušaju neovlaštenog pristupa.
Ovakav pristup je također danas odbačen, niti je kao takav implementiran u sklopu whisker paketa.

3.5. Direktoriji s referencama na sebe

Jedna od novijih tehnika u ovom području bazira se na samo referenciranju direktorija. Naime, poznato je da niz ".." predstavlja nad direktorij, a da niz "." označava trenutni direktorij.
To bi ujedno značilo da će niz "c:\temp\.\.\.\.\.\" biti ekvivalentan "c:\temp\" nizu, odnosno upitu koji pristupa /tmp direktoriju (u svijetu Unix operativnog sustava naravno, naveo bi se samo /tmp direktorija, bez labele c:).
Tako na primjer, ukoliko se želi izbjeći podudaranje sa /cgi-bin/phf potpisom IDS sustava, zahtjev bi mogao izgledati ovako nekako /./cgi-bin/./phf.
Za detekciju ovakvog pristupa raw IDS sustavi imaju tri mogućnosti:

- prijavljivanje postojanja /./ niza, te nastavak sa procesiranjem, čime bi se administratora upozorilo na eventualni pokušaj neautoriziranog pristupa,
- ignorirati ovakve pokušaje, što je gotovo neprihvatljivo rješenje, budući da može rezultirati velikim brojem nedetektiranih pokušaja neovlaštenog pristupa web poslužiteljima,
- unošenje logike u ovakve implementacije, kako bi se svi nizovi oblika /./ zamijenili jednostrukim slash znakom, što bi onemogućilo pokušaj zavaravanja. Takav pristup bi naravno nešto usporio sam sustav, no to je cijena koja se mora platiti.

Implementacija ove metode je jedan od razloga zašto unutar whisker paketa nisu implementirane dvije prethodno opisane metode.
Naime, pokazalo se da će kombiniranje metode URL kodiranja sa ovom metodom (-E opcija kod whisker v1.0, -I opcija kod whisker v1.3 paketa) rezultirati uspješnim zavaravanjem trenutno najpopularnijih i najkvalitetnijim IDS sustava, što uklanja potrebu za prethodno opisanim metodama, budući da su za njih implementirane metode detekcije.
No, kako su uspješno provedeni prvi pokušaji neovlaštenog pristupa korištenjem ove metode, proizvođači kvalitetnijih IDS sustava su pribjegli implementaciji rješenja ovakvih napada, što i ovaj napad čini sve manje upotrebljivim.

3.6. Prerano završavanje zahtjeva

Ovakav pristup prvenstveno je namijenjen zaobilaženju detekcije od strane smart IDS sustava.
Naime, kod smart sustava često se u svrhu ubrzanja rada, odnosno u svrhu procesiranja sve veće količine HTTP zahtjeva, koristi nešto blaži postupak procesiranja paketa, što ih automatski čini ranjivim na određenu grupu napada.

Pristup koji se kao relevantni pokazatelj malicioznog ili regularnog prometa uzima samo zahtjev klijenta (npr. `GET /cgi-bin/ime_skripte.cgi HTTP/1.0`), a da se podatci koji dalje slijede jednostavno ignoriraju.
Na primjer tipični zahtjev klijenta izgledao bi ovako:

```
GET /some.file HTTP/1.0\r\n
Header: blah \r\n
Header: blah \r\n
Header: blah \r\n
Header: blah \r\n
\r\n
```

Proizvođači određene skupine IDS sustava smatra da je nepotrebno dalje ispitivati brojne Header parametre, ukoliko je sam zahtjev regularan, kako bi na taj način ubrzali rad svojih sustava.
Takovi sustavi bi stali sa pretraživanjem potpisa nakon HTTP/1.0\r\n niza, koji opisuje korištenu verziju protokola.
No, pritom se mora biti pažljiv. Zahtjev oblika:

```
GET /%20HTTP/1.0%0d%0aHeader:%20/../../cgi-bin/some.cgi
HTTP/1.0\r\n\r\n
```

koji predstavlja regularan HTTP zahtjev, a ujedno se prevodi u zahtjev oblika:

```
GET / HTTP/1.0\r\nHeader: /../../cgi-bin/some.cgi HTTP/1.0\r\n\r\n
```

ili ljepše formirano:

```
GET / HTTP/1.0\r\n
Header: /../../cgi-bin/some.cgi HTTP/1.0\r\n
\r\n
```

U ovom slučaju IDS sustav bi stao sa pretraživanjem potpisa unutar pristiglog paketa nakon prvog, i u svrhu napada umetnutog 'lažnog' završetka, što bi rezultiralo zaobilaženjem sigurnosnih mjera ove skupine IDS sustava.
Ukoliko se pretpostavi da će IDS sustav prvo obaviti URL dekodiranje (što je slučaj kod većine njih), analiza zahtjeva završila bi nakon prvog lažno podmetnutog završetka, što bi ovako zamaskiran maliciozan HTTP zahtjev ostavilo neprimjećen.
Ovakav pristup implementiran je unutar whisker v1.3 paketa pod opcijom –l 3.

3.7. Skrivanje parametara

Kada se govori o IDS sustavima, koji spadaju u smart skupinu istih sustava potrebno je svakako spomenuti i pitanja koja se odnose na prosljeđivanje parametara poslužitelju. Tipično prosljeđivanje parametara GET metodom izgledalo bi ovako:

```
somepage.php?name=rfp&prog=whisker&enemy=IDS&param=...
```

Ukoliko se IDS sustavom želi onemogućiti pristup samo dokumentima na web poslužitelju, može se zaključiti da u tom slučaju nema potrebe za analizom na ovaj način proslijeđenih parametara.
Samim time poboljšale bi se performanse sustava, budući da bi se na taj način omogućilo procesiranje veće količine zahtjeva u jedinici vremena.
U tu svrhu neke implementacije prekidaju pretraživanje znakova koji slijede nakon znaka ?, budući da se tretiraju kao korisnički parametri koje nije potrebno analizirati.
No, ovakav pristup se treba pomno razmotriti, budući da iskusniji napadači mogu vrlo lako iskoristiti ovakav pristup u svrhu neovlaštenog pristupa resursima web poslužitelja. Na primjer, slijedeći zahtjev:

```
GET /index.htm%3fparam=/../cgi-bin/ime_skripte.cgi HTTP/1.0
```

poslužitelj bi interpretirao kao valjani zahtjev oblika:

```
GET /index.htm?param=/../cgi-bin/ime_skripte.cgi HTTP/1.0
```

što bi također moglo zavarati IDS sustave.
Za uspješno provođenje ovakvog pristupa također je bitno da IDS sustav prije bilo kakovih analiza obavlja URL dekodiranje.
Ukoliko je to slučaj, takav IDS sustav će ignorirati sadržaj koji slijedi nakon samog znaka ?, što će onemogućiti detekciju /cgi-bin/ime_skripte.cgi potpisa, kojeg se želi uočiti.
Rješenje ovog problema je da se prvo iz kompletnog zahtjeva izdvoji dio koji se želi analizirati, a tek da se onda obavi potrebno dekodiranje. Na taj način bi se bez ikakovih problema uočio traženi potpis unutar ovakvog malicioznog HTTP zahtjeva.

3.8. Krivo formiranje HTTP zahtjeva

Prema RFC specifikaciji HTTP protokola (verzija 1.0), regularan HTTP zahtjev izgleda ovako:

```
Method <space> URI <space> HTTP/ Version CRLF CRLF
```

Ono što je potrebno posebno napomenuti kod ovakve definicije HTTP zahtjeva, je to da su svi elementi međusobno odijeljeni jednim razmakom, te da se moraju pojavljivati u ovakvom specifikacijom određenom redoslijedu.
To znači da se na vrlo jednostavan način mogu izdvojiti pojedina polja iz jednog ovakvo formiranog zahtjeva, te da se na temelju njih mogu provesti odgovarajuće akcije.
U ovom pogledu zanimljive su implementacije poznatog web poslužitelja za Linux operativni sustav, Apache. Naime, implementacije Apache web poslužitelja nakon inačice 1.3.6. dopuštaju formiranje HTTP zahtjeva slijedećeg oblika:
`Method <tab> URI <tab> HTTP/ Version CRLF CRLF`
Prihvaćanje ovakvih zahtjeva može u velikoj mjeri narušiti funkcionalnost većine današnjih IDS sustava, koji svoj rad baziraju isključivo na RFC specifikaciji, te ne predviđaju ovakvu mogućnost kreiranja HTTP upita.
Također treba imati na umu v0.9 specifikaciju koja dozvoljava upite oblika:

```
GET <space> URI CRLF
```

što također može prevariti IDS sustave koji su bazirani isključivo na HTTP 1.0 verziji ovog protokola, budući da takvi očekuju tri elementa kod svakog HTTP upita.
Whisker v1.3 programski paket implementira mogućnost kreiranja upita sa tab separatorom (-l 6 opcija), dok ne posjeduje mogućnosti kreiranja zahtjeva baziranih na 0.9 verziji HTTP protokola. No, postoji mogućnost dodavanja koda kojim će se vrlo jednostavno i to postići.

3.9. Dugačke URL adrese

Mnoge implementacije raw IDS sustava, u svrhu poboljšanja performansi pregledavaju samo prvih xx bajtova svakog HTTP upita, na temelju čega donose odluku o prihvaćanju, odnosno odbacivanju paketa.
To je posve pravilan pristup, budući da u normalnim okolnostima prvi redak HTTP zahtjeva sadrži URL adresu resursa koji se želi dobaviti.
No, ovakav pristup se također može vrlo jednostavno iskoristiti pažljivim formiranjem dovoljno dugačkih HTTP zahtjeva, kako bi se na taj način izbjegla detekcija određene grupe IDS sustava. Pogledajmo zahtjev slijedećeg oblika:

```
GET /rfprfp<lots of characters>rfprfp/../cgi-bin/some.cgi HTTP/1.0
```

Ideja ovakvog pristupa je da se umetne dovoljna količina proizvoljnih znakova, koja će originalni upit pomaknuti van područja upita koji je uključen u IDS analizu.
Jedini nedostatak ovakve metode zaobilaženja detekcije je taj što će takvi vrlo dugački upiti rezultirati uočljivim bilježenjem logova na poslužitelju, što može otkriti napad.
Whisker paket implementira i ovu mogućnost, a inicijalno umeće 1-2K proizvoljnih znakova, što udovoljava većini potreba, Količina dodanih znakova kontrolira se XXIDSMode4Limit varijablom, tako da se mogu definirati i drugačije vrijednosti ovog parametra ukoliko zaista postoji potreba.

3.10. DOS/Win sintaksa

Poznato je da DOS/Win implementacije koriste u svrhu razdvajanja direktorija znak \ , samo zato što Unix implementacije u tu svrhu koriste / znak.
Ukoliko se promotri specifikacija HTTP protokola, uočit će se da tamo definirana pravila također uključuju znak /, što je suprotno od pristupa koji se koristi kod Microsoftovih poslužitelja.
Upravo zato Microsoft u posljednje vrijeme sve više prelazi na protokolom definiranu sintaksu, koja se implementira interno unutar IIS poslužitelja.
No zbog kompatibilnost s ranijim inačicama ovog poslužitelja, još uvijek je dozvoljeno korištenje stare sintakse. To ujedno znači da su kod Microsoftovih poslužitelja dozvoljeni upiti oblika:

```
/cgi-bin\some.cgi
```

što će također promaknuti većini IDS implementacija. Jedino o čemu je potrebno posebno voditi računa kod ovakvog pokušaja napada je to da prvi znak mora obavezno biti znak /, budući da obrnuto neće funkcionirati.
Unutar whisker paketa u tu svrhu koristi se –l 8 opcija.

3.11. Korištenje NULL znaka

Kako je poznato, biblioteke funkcija C programskog jezika, za označavanje kraja znakovnog niza koriste NULL znak.
Iako je većina današnjih IDS sustava prespora da bi koristila ovakav pristup, odnosno spomenute biblioteke funkcija, korištenje NULL znaka za označavanje kraja znakovnog niza u praksi je vrlo često. Takav pristup se u ovom slučaju također može lukavo iskoristiti.
U tu svrhu zamislimo sljedeći HTTP zahtjev:

```
GET%00 /cgi-bin/ime_skripte.cgi HTTP/1.0
```

Prilikom procesiranja ovakvog zahtjeva, scenarij je slijedeći:

- web poslužitelj prima ovako formirani maliciozni zahtjev, te odvaja Method i URL elementa u svrhu njihovog zasebnog analiziranja.
- slijedi zasebno URL dekodiranje prvo elementa Method, a zatim i URL adrese, ili obrnuto
- na ovaj način formirano polje Method biti će valjano iako na kraju sadrži NULL znak, te će ga kao takvog web poslužitelj i tretirati.
- IDS sustav za razliku od poslužitelja dekodira odjednom cijeli HTTP zahtjev
- primjenjujući funkcije za rad s znakovnim nizovima, u svrhu njihovog analiziranja, IDS sustav nailazi na NULL znak, te na tom mjestu prekida s provođenjem sigurnosnih provjera, što će omogućiti zaobilaženje sigurnosnih mjera IDS sustava.

Napomena: Apache web poslužitelj neće prihvatiti niti jedan zahtjev koji u sebi sadrži %00 (NULL) znak ili %2f (/)znak, što će onemogućiti uspješno provođenje ovog tipa napada. Ne smije se zaboraviti važna činjenica, da web poslužitelj maskirani maliciozni zahtjev još uvijek mora interpretirati kao valjani HTTP upit.
Unutar whisker paket ova metoda implementirana je –l 0 opcijom.

3.12. Velika i mala slova

Karakteristika DOS/Win datotečnog sustava, u odnosu na Unix datotečni sustav, je ta da isti nije osjetljiv na velika i mala slova. To znači da će zahtjevi tipa INDEKS.HTML, indeks.html, iNdEkS.HtMl biti tretirani potpuno jednako.
Ova posebnost se također može iskoristiti za zaobilaženje sigurnosnih mjera IDS sustava, budući da zahtjev /cgi-bin/ime_skripte.cgi, kojeg IDS sustav prepoznaje neće odgovarati zahtjevu oblika /CGI-BIN/IME_SKRIPTE.CGI.
U svrhu zaobilaženja detekcije, anti IDS sustavi u tu svrhu mogu koristiti kombinaciju velikih i malih slova u svrhu uspješnijeg zavaravanja IDS sustava.
Whisker paket implementira ovu metodu korištenjem –l 7 opcije.

3.13. Rastavljanje mrežne komunikacije

Ova metoda je trenutno jedina postojeća metoda kod anti IDS sustava, koja radi na mrežnom OSI sloju, a ne na aplikacijskom, kao što je to slučaj sa prethodno opisanim metodama.
Većina postojećih implementacija IDS sustava pretražuje potpise unutar samo jednog paketa. Ne postoji mogućnost rastavljanja potpisa i njihovog uspoređivanja sa više paketa, što se također može iskoristiti kao sredstvo zaobilaženja sigurnosnih mjera IDS sustava.
Whisker anti IDS sustav koristi ovo svojstvo rastavljanjem zahtjeva i njegovim slanjem unutar odvojenih IP paketa.
Odmah treba napomenuti da ovakav pristup nema nikakve veze sa IP fragmentacijom paketa, budući da se jednostavno radi o više odvojenih paketa, od kojih svaki u svom podatkovnom segmentu sadrži dijelove originalnog HTTP upita.
Na primjer, GET / HTTP/1.0 zahtjev može se razložiti na više dijelova, koji bi se onda slali u zasebnim IP paketima. Svaki paket bi u tom slučaju u svom podatkovnom dijelu sadržavao dio upita (npr. "GE", "T ", "/", " H", "T", "TP", "/1", ".0".), čime bi se zavarao IDS sustav.
Rješenje ovog problema je da se prvo prikupe podaci koji pripadaju jednom komunikacijskom kanalu, a da se tek naknadno provode sigurnosne analize. No, ovakav pristup zahtjeva vrlo dobro poznavanje samog protokola, te poznavanje definicije pojma sesija unutar HTTP protokola.

4. Zaključak

Unutar ovoga dokumenta opisan je vrlo moćan anti IDS programski alat, koji omogućuje zaobilaženje sigurnosnih mjera koje provode IDS sustavi. Opisane su osnovne metode koje se u tu svrhu koriste, zajedno sa načinima njihovog korištenja.
Također postoji mogućnost kombiniranja iznesenih metoda, kako bi se na taj način pružilo što više mogućnosti zavaravanja IDS sustava. Sintaksa je:

```
whisker.pl -h www.ime_poslužitelja.com -I 124
```

gdje se naravno mora navesti pravo ime poslužitelja, te parametri koji opisuju koja se metoda želi koristiti. Ovaj primjer koristiti će metode sa oznakama 1, 2, i 4, za pristup poslužitelju sa imenom www.ime_poslužitelja.com .
Također treba napomenuti da whisker alat radi samo za HTTP protokol, odnosno za web servis.
Whisker programski paket može se nabaviti na web stranici http://www.wiretrip.net/rfp/.