Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Rozšírenie nástroja SLAmeter o detekciu anomálií v počítačových sieťach

Diplomová práca

Príloha B

POUŽÍVATEĽSKÁ PRÍRUČKA bmIDS v2.3

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika
Školiace pracovisko:	Ústav výpočtovej techniky (ÚVT)
Vedúci práce:	Ing. Adrián Pekár, PhD.
Konzultant:	Ing. Ján Juhár

Košice 2015 **Bc. Ladislav Berta**

Obsah

Zoznam obrázkov

Zoznam tabuliek

1 Funkcia programu

Systém bmIDS predstavuje nástroj pre detekciu narušenia v počítačových sieťach. Pozostáva z dvoch aplikácií, analyzéra *bmIDSanalyzer* a aplikácie *ids*, ktorá je súčasťou webovej aplikácie SLAmeter.

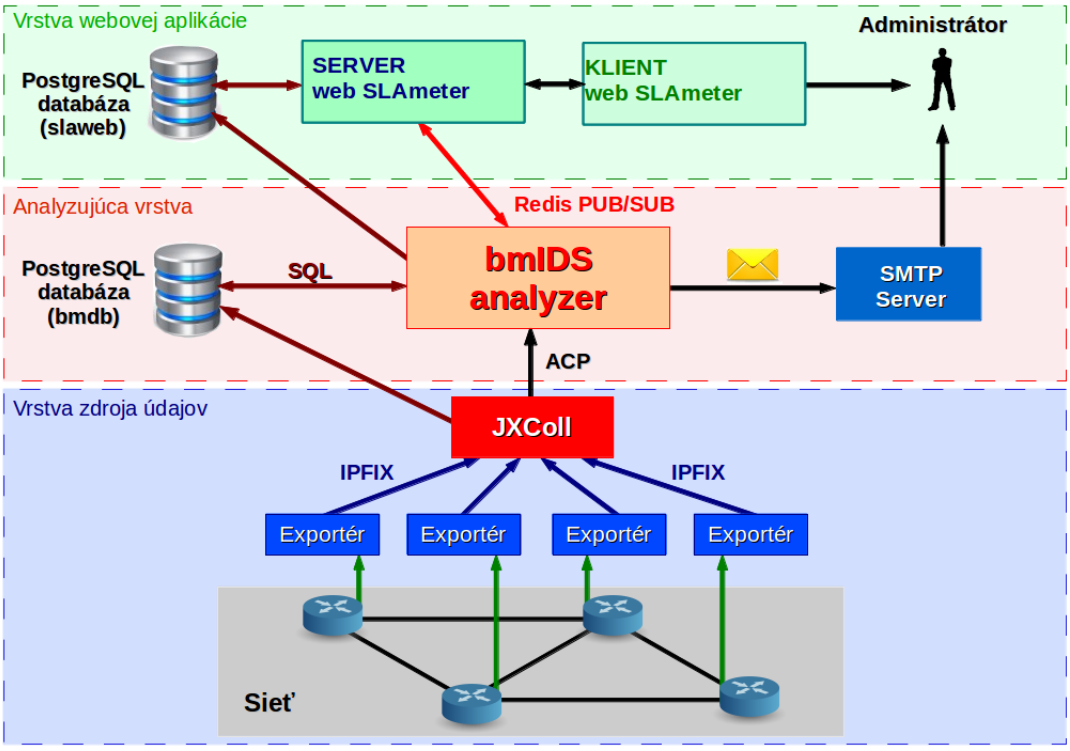
Analyzér *bmIDSanalyzer* je konzolová aplikácia, ktorého úlohou je vyhodnotenie a spracovanie údajov o prebiehajúcej komunikácii v sieti, a na základe výsledkov vyhodnotenia a fuzzy logiky určiť mieru pravdepodobnosti útokov. Zdrojom údajov pre systém bmIDS sú nástroje exportér (mybeem) a kolektor (JXColl), ktoré zabezpečujú monitorovanie siete a poskytujú informácie analyzéru v podobe správ IPFIX protokolu. Prenos správ je zabezpečený aplikačným rozhraním ACP, ktoré správy od kolektora poskytuje v reálnom čase. V režime detekcie je analyzér spustený na serveri kde neustále vyhodnocuje sieťovú prevádzku. V aktuálnom stave systém dokáže detekovať šesť typov útokov:

- útok typu Syn Flood
- útok typu Udp Flood
- útok typu Port Scan
- útok typu Rst Flood
- útok typu TTL expiry Flood
- útok typu Fin Flood

V prípade zaznamenania útokov zapisuje údaje o kritickej prevádzke do databázy. Zároveň môže odosielať vyhodnotené údaje aj webovej aplikácii *ids* ako aj emailové notifikácie pre používateľa s detailami o útoku. Webová aplikácia *ids* pozostáva z dvoch sekcií, zo sekcie *Monitorovania* a sekcia *Záznamy útokov*. Sekcia *Monitorovania* umožňuje používateľovi sledovať vyhodnotenú prevádzku v reálnom čase, pričom sekcia *Záznamy útokov* slúži pre zobrazovanie historických záznamov.

2 Architektúra systému

Hlavným cieľom programu je detekcia útokov v počítačovej sieti s využitím komponentov exportéra a kolektora, ktoré slúžia na exportovanie, zhromažďovanie a následné poskytovanie informácií pre analyzujúcu aplikáciu. Informácie o IP tokoch sú poskytované vo formáte správ IPFIX protokolu, ktorý umožňuje export veľkej množiny informačných elementov, ktoré reprezentujú charakteristiky monitorovanej siete. Zoznam používaných informačných elementov je uvedený v systémovej príručke pre bmIDSv2.2 [5]. Pre detekciu a klasifikáciu sieťovej prevádzky využíva systém bmIDS fuzzy logiku, ktorá pre detekciu anomálií predstavuje jednu z najprogresívnejších metód.



Obr. 2 – 1 Architektúra systému bmIDS

Systém bmIDS tvoria dve samostatné aplikácie. Jednou je analyzujúca aplikácia *bmIDSanalyzer*, ktorá prijíma informácie o sieti, vyhodnocuje ich, a na základe vyhodnotených údajov určuje mieru pravdepodobnosti prienikov. Okrem detekcie,

ktorá je hlavnou úlohou systémov IDS, umožňuje posielanie údajov webovej aplikácii, ktorá tvorí druhú časť systému. Aplikácia *ids* je súčasťou webovej aplikácie nástroja SLAmeter. Webová aplikácia umožňuje pre používateľa zobrazovať vyhodnotené údaje analyzérom v podobe grafických priebehov v reálnom čase. Okrem sledovanie výstupov detekcie umožňuje administrátorovi zobrazovať detailné informácie o narušeniach, ktoré sú v prípade detekovania útokov ukladané do databázy.

Architektúra systému bmIDS je zobrazená na obrázku Obr. 2 – 1. Pozostáva z nasledujúcich vrstiev:

- **Vrstva zdroja údajov** - tvoria ju komponenty Exportér mybeem, ktorý sleduje sieťovú prevádzku, klasifikuje ju do tokov a následne ich exportuje ku kolektoru pomocou protokolu IPFIX. Kolektor JXColl slúži ako zhromažďovač záznamov ktoré sú exportérmi exportované. Zhromaždené záznamy ukladá do databázy *bmdb* a pomocou protokolu ACP ich poskytuje analyzujúcej aplikácii bmIDSanalyzer.
- **Analyzujúca vrstva** - predstavuje ju samotný *bmIDSanalyzer*, ktorý získané údaje zo spodnej vrstvy vyhodnocuje. V režime detekcie využíva PostgreSQL databázu *slaweb* z vrchnej časti architektúry, kde ukladá údaje o kritickej prevádzke. V režime učenia sa pripája na databázu *bmdb*, odkiaľ získava historické záznamy o sieťovej prevádzke pre vytvorenie obrazu štandardnej prevádzky. Vytvorením reálno-časovej komunikácie medzi serverom webovej aplikácie SLAmeter, poskytuje analyzér vyhodnotené údaje pre webovú aplikáciu. Pri zaznamenaní útoku sa analyzér pripojí na smtp server, cez ktorý pošle notifikáciu používateľovi v podobe emailovej správy.
- **Vrstva webovej aplikácie** - predstavuje ju serverová a klientska časť webovej aplikácie SLAmeter, ako aj PostgreSQL databáza *slaweb*. Vyhodnotené údaje o prevádzke poskytuje analyzér v reálnom čase serverovej časti. Tieto údaje

sú ďalej poskytované klientskej časti nástroja, ktorý ich spracuje a následne ich vo webovom prehliadači zobrazí používateľovi v podobe grafov.

2.1 Popis programu bmIDSanalyzer

Program *bmIDSanalyzer* tak ako väčšina systémov pre detekciu narušenia, funguje v dvoch režimoch:

- v režime učenia
- v režim detekcie

2.1.1 Režim učenia

V tomto režime sa vytvára obraz štandardnej prevádzky na základe historických záznamov. Zdrojom údajov je databáza bmdb, do ktorej sú kolektorom ukladané záznamy o sieťovej prevádzke. Bližší popis režimu učenia je dostupný v príslušnej systémovej príručke [5].

2.1.2 Režim detekcie

Základom režimu detekcie je fuzzy logika, a fuzzy subsystémy. Vstupmi do týchto subsystémov sú príslušné charakteristiky pre jednotlivé typy útokov. Výstupom je pravdepodobnosť prieniku konkrétneho typu útoku. V prípade zaznamenania útokov sú údaje o prieniku ukladané do databázy webového servera *slaweb*. Používateľ má možnosť tieto údaje prezerať vo webovom rozhraní aplikácie. Pre upozornenie administrátora o detekovanom prieniku, môže analyzér posielať notifikácie s detailami o útoku v podobe emailových správ na účet používateľa. Emailová adresa sa zadáva z konfiguračného súboru ako aj možnosť voľby posielania respektíve neposielania notifikácií.

Systém je schopný v režime detekcie o určenie prieniku nasledujúcich útoku:

1. útok typu SYN Flood – záplava TCP paketmi so SYN príznakom
2. útok typu UDP Flood – záplava TCP paketmi protokolu UDP

3. útok typu Port Scan – skenovanie portov
4. útok typu RST Flood – záplava TCP paketmi s RST príznakom
5. útok typu TTL expiry Flood – záplava ICMP paketmi
6. útok typu FIN Flood – záplava TCP paketmi s FIN príznakom

Pre bližší popis režimu učenia je potrebné pozrieť príslušnú systémovú príručku [5].

2.1.3 Popis webovej aplikácie *ids*

Aplikáciu ids tvoria dve sekcie. Sekcia *Monitoring* je určená pre používateľa a poskytuje možnosť sledovať výstupy detekcie v reálnom čase. Druhá sekcia *Attack logs* umožňuje zobrazovanie detailných informácií o narušeniach z rôzneho časového obdobia. Údaje kritickej prevádzke sa získavajú z databázy webového servera, kam ich analyzujúca aplikácia bmIDSanalyzer ukladá. Aplikácia *ids* je implementovaná vo webovej aplikácii nástroja SLAmeter. Architektúra tohto nástroja je podrobne popísaná v príručke webovej aplikácie SLAmeter [4].

Strana servera webovej aplikácie nástroja SLAmeter je realizovaná ako Django aplikácia. Sú v nej implementované moduly, ktoré predstavujú hlavnú súčasť aplikácie ids. Ich úlohou je prijímanie požiadaviek od webového klienta. Moduly sú na základe konfiguračného súboru rozdelené do dvoch sekcií. Pričom moduly sekcie Monitoring komunikujú s klientom webovej aplikácie cez služby Websocket protokolu, modul sekcie Attack logs je definovaná ako REST služba. Tento modul používa komunikáciu s databázou *slaweb* a získané hodnoty sprístupňuje pre webového klienta v podobe REST služieb. Komunikácia modulov sekcie Monitoring s analyzujúcou aplikáciou je zabezpečená pomocou služby Publish/Subscribe technológie Redis. Komunikácia medzi serverom a analyzérom teda prebieha cez kanáli Redis. Prijaté údaje od analyzujúcej aplikácie sú poskytované klientovi, a cez riadič modulu sekcie Monitoring poskytované modulom zobrazujúce grafické priebehy. Na základe pri-

jatých údajov sa realizuje aktualizácia grafov pre vykreslenie sledovaných charakteristík a miery podozrenia jednotlivých typov útokov. Komunikáciu, zobrazovanie, aktualizovanie a jednotlivé akcie na stránke zabezpečujú riadiče jednotlivých modulov, k nim prislúchajúce šablóny a komponenty grafov. Jednotlivé triedy sú popísané v nasledujúcich kapitolách.

3 Popis tried, členských premenných a metód

Nakoľko táto práca sa venuje rozšíreniu nástroja bmIDS a implementovaniu aplikácie ids v rámci webovej aplikácie nástroja SLAmeter, v tejto kapitole budú uvedené iba triedy, moduly a riadiče, ktoré boli vytvorené pre dosiahnutie rozšírenia, respektíve k vytvoreniu webovej aplikácie ids. Popis ostatných tried a metód je uvedený v príslušnej systémovej príručke [5].

3.1 Triedy balíka *sk.tuke.cnl.bmIDS.processor:*

Balík *sk.tuke.cnl.bmIDS.detector* obsahuje všetky triedy procesorov pre jednotlivé typy útokov a triedu *TrafficProcessor* od ktorej dedia.

3.1.1 Trieda RstFloodProcessor

Trieda RstFloodProcessor predstavuje vlákno pre spracovanie a klasifikovanie tokov podľa charakteristík útoku typu RST záplava.

Konštruktor:

`public RstFloodProcessor()`

Bezparametrický konštruktor pre režim učenia.

`public RstFloodProcessor(RstFloodDetector rstFloodDetector)`

Konštruktor pre režim detekcie.

Metódy:

`public void run()`

Hlavná metóda ktorá spusti prijímanie a spracovanie tokov.

`private void processTraffic()`

Spracováva prevádzku podľa charakteristík útoku typu RST záplava.

`private void classifyTraffic()`

Klasifikuje IP toky podľa charakteristík útoku typu RST záplava.

`private void evaluateTraffic(int reason)`

Spúšťa vyhodnocovanie zatriedenej prevádzky podľa charakteristík útoku typu záplava RST paketmi.

Parameter:

int reason – dôvod vyhodnotenia (môže byť normálne alebo agresívne)

`public int learn(IPFlow ipFlow)`

Spracuje IP tok v režime učenia.

Parameter:

IPFlow ipFlow – IP tok z databázy v režime učenia

Návratová hodnota:

Maximálny počet RST príznakov po vyhodnotení, resp. 0 ak prebehlo iba spracovanie

3.1.2 Trieda TTLFloodProcessor

Reprezentuje vlákno pre spracovanie tokov podľa charakteristík útoku typu TTL expiry záplava.

Konštruktor:

`public TTLFloodProcessor()`

Bezparametrický konštruktor pre režim učenia.

`public TTLFloodProcessor(TTLFloodDetector ttlFloodDetector)`

Konštruktor pre režim detekcie.

Metódy:

`public void run()`

Hlavná metóda ktorá spusti prijímanie a spracovanie tokov.

`private void processTraffic()`

Spracováva prevádzku podľa charakteristík útoku typu TTL Expiry záplava.

`private void classifyTraffic()`

Klasifikuje IP toky podľa charakteristík útoku typu TTL Expiry záplava.

`private void evaluateTraffic(int reason)`

Spúšťa vyhodnocovanie zatriedenej prevádzky podľa charakteristík útoku typu záplava TTL Expiry paketmi.

Parameter:

int reason – dôvod vyhodnotenia

`public int learn(IPFlow ipFlow)`

Spracuje IP tok v režime učenia.

Parameter:

IPFlow ipFlow – IP tok z databázy v režime učenia

Návratová hodnota:

maximálny počet ICMP paketov (typ=11, kod=0) po vyhodnotení, resp. 0 ak prebehlo iba spracovanie

3.1.3 Trieda FinFloodProcessor

Trieda FinFloodProcessor predstavuje vlákno pre spracovanie tokov podľa charakteristík útoku typu FIN záplava.

Konštruktor:

`public FinFloodProcessor()`

Bezparametrický konštruktor pre režim učenia.

`public FinFloodProcessor(FinFloodDetector finFloodDetector)`

Konštruktor pre režim detekcie.

Metódy:

`public void run()`

Hlavná metóda ktorá spusti prijímanie a spracovanie tokov.

`private void processTraffic()`

Spracováva prevádzku podľa charakteristík útoku typu FIN záplava.

`private void classifyTraffic()`

Klasifikuje IP toky podľa charakteristík útoku typu FIN záplava.

`private void evaluateTraffic(int reason)`

Spúšťa vyhodnocovanie zatriedenej prevádzky podľa charakteristík útoku typu záplava FIN paketmi.

Parameter:

int reason – dôvod vyhodnotenia

`public int learn(IPFlow ipFlow)`

Spracuje IP tok v režime učenia.

Parameter:

IPFlow ipFlow – IP tok z databázy v režime učenia.

Návratová hodnota:

Maximálny počet FIN príznakov po vyhodnotení, resp. 0 ak prebehlo iba spracovanie

3.2 Triedy balíka *sk.tuke.cnl.bmIDS.detector:*

Balík *sk.tuke.cnl.bmIDS.detector* obsahuje všetky triedy detektorov pre jednotlivé typy útokov a triedu *FuzzyDetector* od ktorej dedia.

3.2.1 Trieda RstFloodDetector

Vyhodnocuje roztriedenú prevádzku voči útoku typu RST záplava.

Metódy:

`public RstFloodDetector()`

Vytvorí objekt detektora a načíta fuzzy údaje zo súboru.

Hádže:

Fuzzy Control error: Can't load fcl file – ak nevie načítať fcl súbor ktorý obsahuje fuzzy pravidlá.

`public synchronized void setLimits(int maxRstCount)`

Vytvorí obraz štandardnej prevádzky v podobe fuzzy množín podľa naučenej maximálnej hodnoty sledovanej charakteristiky.

Návratová hodnota:

maxRstCount – maximálny počet paketov s RST príznakom v tokoch s rovnakou cieľovou IP adresou a rovnakým cieľovým portom

`public int getMaxRstCount()`

Vráti maximálnu hodnotu najpravejšej fuzzy množiny.

Návratová hodnota:

Maximálnu hodnotu najpravejšej fuzzy množiny.

`public boolean isWebClientConnected()`

Vracia príznak podľa toho, či je webový klient pripojený.

Návratová hodnota:

true – ak je pripojený klient

false – ak klient nie je pripojeny

`public void setWebClientConnected(boolean flag)`

Nastavuje príznak pripojenia webového klienta.

Parameter:

boolean flag – príznak nastavenia

`public void evaluate(LinkedList<RstFloodSignTraffic> rstFTrafficList, int reason)`

Vyhodnotí roztriedenú prevádzku fuzzy podsystémom.

Parameter:

rstFTrafficList – roztriedená prevádzka podľa charakteristík útoku typu RST záplava

reason – dôvod vyhodnotenia na zákalde čoho vyhodnocuje prevádzku, môže byť normálne alebo agresívne

`private int getPastRstCount()`

Vráti priemerný počet RST príznakov v predchádzajúcich 3 cykloch.

Návratová hodnota:

Priemerný počet RST príznakov v predchádzajúcich 3 cykloch.

`private void updatePastPacketCount(int rstCount)`

Aktualizuje zoznam vyhodnotenej prevádzky v predchadzajúcich troch cykloch.

Parameter:

rstCount – počet RST príznakov

`private void sendOutputToClient(Date time, int rstCount, double probability)`

Odošle údaje o vyhodnotenej prevádzke webovej aplikácii.

Parameter:

time – čas vyhodnotenia, *rstCount* – hodnota sledovanej charakteristiky útoku typu RST záplava, *probability* – pravdepodobnosť výskytu útoku.

`private void saveOutputToDatabase()`

Uloží údaje o útoku do databázy.

`private void insertAttackDataToDB()`

Vloží údaje o novom útoku do tabuľky v databáze.

`private void updateAttackDataInDB()`

Aktualizuje údaje o útoku v hlavnej tabuľke a vloží údaje o útoku do detailnej tabuľky.

3.2.2 Trieda TTLFloodDetector

Vyhodnocuje roztriedenú prevádzku voči útoku typu TTL Expiry záplava.

Metódy:

`public TTLFloodDetector()`

Vytvorí objekt detektora a načíta fuzzy údaje zo súboru.

Hádže:

Fuzzy Control error: Can't load fcl file – ak nevie načítať fcl súbor ktorý obsahuje fuzzy pravidlá.

`public synchronized void setLimits(int maxTtlCount)`

Vytvorí obraz štandardnej prevádzky v podobe fuzzy množín podľa naučenej maximálnej hodnoty sledovanej charakteristiky.

Návratová hodnota:

maxTtlCount – maximálny počet ICMP paketov (typ=11, kod=0) v tokoch s rovnakou cieľovou IP adresou

`public int getMaxPacketCount()`

Vráti maximálnu hodnotu najpravejšej fuzzy množiny.

Návratová hodnota:

Vracia maximálnu hodnotu najpravejšej fuzzy množiny.

`public boolean isWebClientConnected()`

Vracia príznak podľa toho, či je webový klient pripojený.

Návratová hodnota:

true – ak je pripojený klient

false – ak klient nie je pripojeny

`public void setWebClientConnected(boolean flag)`

Nastavuje príznak pripojenia webového klienta.

Parameter:

boolean flag – príznak nastavenia

`public void evaluate(LinkedList<TTLFloodSignTraffic> ttlFTrafficList, int reason)`

Vyhodnotí roztriedenú prevádzku fuzzy podsystémom.

Parameter:

ttlFTrafficList – roztriedená prevádzka podľa charakteristík útoku typu TTL Expiry záplava

reason – dôvod vyhodnotenia na zákalde čoho vyhodnocuje prevádzku, môže byť normálne alebo agresívne

`private int getPastPacketCount()`

Vráti priemerný počet ICMP paketov v predchádzajúcich 3 cykloch.

Návratová hodnota:

Priemerný počet RST príznakov v predchádzajúcich 3 cykloch.

`private void updatePastPacketCount(int ttlCount)`

Aktualizuje zoznam vyhodnotenej prevádzky v predchadzajúcich troch cykloch.

Parameter:

ttlCount – počet ICMP paketov (typ=11, kód=0)

`private void sendOutputToClient(Date time, int ttlCount,`

`double probability)`

Odošle údaje o vyhodnotenej prevádzke webovej aplikácii.

Parameter:

time – čas vyhodnotenia, *ttlCount* – hodnota sledovanej charakteristiky útoku typu TTL Expiry záplava, *probability* – pravdepodobnosť výskytu útoku.

`private void saveOutputToDatabase()`

Uloží údaje o útoku do databázy.

`private void insertAttackDataToDB()`

Vloží údaje o novom útoku do tabuľky v databáze.

`private void updateAttackDataInDB()`

Aktualizuje údaje o útoku v hlavnej tabuľke a vloží údaje o útoku do detailnej tabuľky.

3.2.3 Trieda FinFloodDetector

Vyhodnocuje roztriedenú prevádzku voči útoku typu FIN záplava.

Metódy:

`public FinFloodDetector()`

Vytvorí objekt detektora a načíta fuzzy údaje zo súboru.

Hádže:

Fuzzy Control error: Can't load fcl file – ak nevie načítať fcl súbor ktorý obsahuje fuzzy pravidlá.

`public synchronized void setLimits(int maxFinCount)`

Vytvorí obraz štandardnej prevádzky v podobe fuzzy množín podľa naučenej maximálnej hodnoty sledovanej charakteristiky.

Návratová hodnota:

maxFinCount – maximálny počet paketov s FIN príznakom v tokoch s rovnakou cieľovou IP adresou a rovnakým cieľovým portom

`public int getMaxFinCount()`

Vráti maximálnu hodnotu najpravejšej fuzzy množiny.

Návratová hodnota:

Vracia maximálnu hodnotu najpravejšej fuzzy množiny.

`public boolean isWebClientConnected()`

Vracia príznak podľa toho, či je webový klient pripojený.

Návratová hodnota:

true – ak je pripojený klient

false – ak klient nie je pripojeny

`public void setWebClientConnected(boolean flag)`

Nastavuje príznak pripojenia webového klienta.

Parameter:

boolean flag – príznak nastavenia

`public void evaluate(LinkedList<FinFloodSignTraffic> finFTrafficList, int reason)`

Vyhodnotí roztriedenú prevádzku fuzzy podsystémom.

Parameter:

rstFTrafficList – roztriedená prevádzka podľa charakteristík útoku typu FIN záplava

reason – dôvod vyhodnotenia na zákalde čoho vyhodnocuje prevádzku, môže byť normálne alebo agresívne

`private int getPastFinCount()`

Vráti priemerný počet FIN príznakov v predchádzajúcich 3 cykloch.

Návratová hodnota:

Priemerný počet FIN príznakov v predchádzajúcich 3 cykloch.

`private void updatePastPacketCount(int finCount)`

Aktualizuje zoznam vyhodnotenej prevádzky v predchadzajúcich troch cykloch.

Parameter:

finCount – počet FIN príznakov

`private void sendOutputToClient(Date time, int finCount, double probability)`

Odošle údaje o vyhodnotenej prevádzke webovej aplikácii.

Parameter:

time – čas vyhodnotenia, *finCount* – hodnota sledovanej charakteristiky útoku typu FIN záplava, *probability* – pravdepodobnosť výskytu útoku.

`private void saveOutputToDatabase()`

Uloží údaje o útoku do databázy.

`private void insertAttackDataToDB()`

Vloží údaje o novom útoku do tabuľky v databáze.

`private void updateAttackDataInDB()`

Aktualizuje údaje o útoku v hlavnej tabuľke a vloží údaje o útoku do detailnej tabuľky.

3.3 Triedy balíka *sk.tuke.cnl.bmIDS.traffic:*

Balík *sk.tuke.cnl.bmIDS.traffic* obsahuje triedy so špecifickými charakteristikami pre jednotlivé typy útokov. Uvedená je trieda RstFloodSignTraffic. Popis metód a atribútov tried TTLFloodSignTraffic a FinFloodSignTraffic ako aj ostatných tried je uvedený v prílohe na CD vo formáte JavaDoc.

3.3.1 Trieda RstFloodSignTraffic

Táto trieda predstavuje roztriedená prevádzku podľa charakteristík útoku typu RST záplava.

Konštruktor:

`public RstFloodSignTraffic()(int obsPoint, String sourceIP, String destIP, int destPort, int rstCount, long since, long till)`

Vytvorí objekt prevádzky podľa charakteristík útoku typu RST záplava.

Parameter:

obsPoint – identifikátor pozorovacej domény, *sourceIP* – zdrojová IP adresa toku, *destIP* – cieľová IP adresa toku, *destPort* – cieľový port toku, *rstCount* – počet RST príznakov toku, *sinceMilis* – začiatok toku v milisekundách, *tillMilis* – koniec toku v milisekundách.

Metódy:

`public String getSrcIP()`

Vracia zdrojovú IP adresu zatriedeného toku.

Návratová hodnota:

srcIP – zdrojová IP adresa zatriedeného toku

`public String getDestIP()`

Vracia cieľovú IP adresu zatriedeného toku.

Návratová hodnota:

destIP – zdrojová IP adresa zatriedeného toku

`public int getDestPort()`

Vracia cieľový port zatriedeného toku.

Návratová hodnota:

destPort – cieľový port zatriedeného toku

`public int getObsPoint()`

Vracia identifikátor pozorovacej domény zatriedeného toku.

Návratová hodnota:

obsPoint – identifikátor pozorovacej domény zatriedeného toku.

`public int getRstCount()`

Vracia počet RST príznakov zatriedeného toku.

Návratová hodnota:

rstCount – počet RST príznakov zatriedeného toku.

`public long getSinceMilis()`

Vracia začiatok zatriedeného toku v milisekundách.

Návratová hodnota:

sinceMilis – začiatok toku zatriedeného v milisekundách.

`public long getTillMilis()`

Vracia koniec zatriedeného toku v milisekundách.

Návratová hodnota:

tillMilis – koniec zatriedeného toku v milisekundách.

`public long getId()`

Vracia ID zatriedeného toku.

Návratová hodnota:

id – ID zatriedeného toku

`public void setId(long id)`

Nastaví ID zatriedeného toku.

`public double getAttackProbability()`

Vracia pravdepodobnosť útoku.

Návratová hodnota:

attackProbability – pravdepodobnosť útoku

`public void setAttackProbability(double attackProbability)`

Nastaví pravdepodobnosť útoku.

`public int edit(IPFlow currentIPFlow)`

Upraví charakteristiky prevádzky podľa hodnôt nového toku.

Parameter:

currentIPFlow – aktuálny nový tok.

Návratová hodnota:

rstCount – počet RST príznakov v zatriedenej prevádzke.

`public void update(TTLFloodSignTraffic traffic, double probability)`

Aktualizuje charakteristiky roztriedenej prevádzky.

Parameter:

TTLFloodSignTraffic traffic – objekt roztriedenej prevádzky.

double probability – pravdepodobnosť útoku.

3.4 Triedy balíka *sk.tuke.cnl.bmIDS.client:*

Balík *sk.tuke.cnl.bmIDS.traffic* obsahuje triedy s údajmi pre vykreslenie grafov v sekcii monitoring, ktoré pochádzajú od analyzéra. Uvedená je trieda RstFloodCharts ktorá obsahuje údaje o vyhodnotenej prevádzke voči útoku typu RST záplava prijaté od analyzéra pre vykreslenie grafov. Popis metód a atribútov tried TTLFloodCharts a FinFloodCharts ako aj ostatných tried je uvedený v prílohe na CD vo formáte JavaDoc.

3.4.1 Trieda RstFloodCharts

Trieda obsahuje údaje o vyhodnotenej prevádzke voči útoku typu RST záplava prijaté od analyzéra pre vykreslenie grafov.

Metódy:

`public boolean isOnPage()`

Vráti hodnotu true alebo false v závislosti od toho, či grafy s novými hodnotami pre útok typu RST záplavú sú, resp. nie sú zobrazené na stránke.

Návratová hodnota:

onPage – vráti s príslušným nastavením, true, ak sú grafy pre útok typu RST záplava aktualizované na stránke, inak false.

`public void setOnPage(boolean onPage)`

Nastaví príznak aktualizovania grafov typu RST záplava na stránke podľa hodnoty argumentu.

Parameter:

onPage – príznak zobrazenie, či sú grafy s novými hodnotami zobrazené.

`public void RstFloodCharts(int maxRstCount)`

Inicializuje grafy pre útok typu RST záplava nastavením hodnôt osi.

Parameter:

maxRstCount – maximálny počet RST príznakov naučenej štandardnej prevádzky.

`private JFreeChart createTrafficChart(final XYDataset dataset,`
`int maxRstCount)`

Vytvorí graf pre zobrazovanie hodnôt parametrov pre útok typu RST Flood v závislosti na čase.

Parameter:

dataset – množina dát grafu.

maxRstCount – maximálny počet RST príznakov naučenej štandardnej prevádzky.

Návratová hodnota:

result – objekt grafu

`private JFreeChart createProbChart(TimeSeriesCollection dataset2)`

Vytvorí graf pre zobrazovanie miery podozrenia pre útok typu RST Flood v závislosti na čase.

Parameter:

dataset2 – množina dát grafu.

Návratová hodnota:

result – objekt grafu

`public void updateCharts(Date date, int rstCount, double prob)`

Aktualizuje hodnoty v grafoch pre útok typu RST Flood.

Parameter:

date – čas útoku.

rstCount – sledovaný parameter.

prob – pravdepodobnosť útoku.

`public byte[] getTrafficChart()`

Vráti graf sledovaného parametra pre útok typu RST záplava.

Návratová hodnota:

chart – graf sledovaného parametra pre útok typu RST záplava.

`public byte[] getProbabilityChart()`

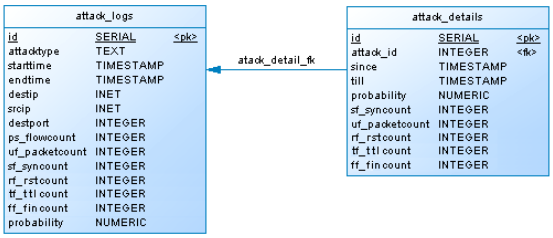
Vráti graf pravdepodobnosti útoku typu RST záplava.

Návratová hodnota:

chart – graf pravdepodobnosti útoku typu RST záplava.

3.5 Popis databázových štruktúr

Systém bmIDS používa databázový server Postgresql na ukladanie údajov o útokoch. Obsahuje dve tabuľky attack logs a attack details. Prvá z nich obsahuje všeobecné údaje o útoku (dátum, čas, typ útoku, namerané hodnoty príslušných charakteristík, IP adresy komunikujúcich strán, porty, pravdepodobnosť útoku) zatiaľ čo druhá slúži pre detailnejšie informácie pri dlhšie trvajúcich útokoch. Dátový model je uvedený na obrázku Obr. 7-1. Na vytvorenie tabuliek slúži skript dbscript.sql, ktorý sa nachádza v prílohe na CD.



Obr. 3 – 1 Dátový model

4 Vstupné súbory

4.1 FCL súbory

Ďalšími vstupnými súbormi sú FCL súbory, ktoré obsahujú popis jednotlivých fuzzy subsystémov v jazyku Fuzzy Control Language (IEC, 1997). Tie sú využívané knižnicou *jFuzzyLogic*, ktorá umožňuje vykonávať jednotlivé operácie na fuzzy množinách. Pre každý typ útoku existuje samostatný FCL súbor, ktorý sa načíta pri spustení programu.

Zoznam používaných FCL súborov:

- ps.fcl - súbor pre útok typu Port Scan
- synf.fcl - súbor pre útok typu SYN Flood
- udpf.fcl - súbor pre útok typu UDP Flood
- rstf.fcl - súbor pre útok typu RST Flood
- ttlf.fcl - súbor pre útok typu TTL expiry Flood
- finf.fcl - súbor pre útok typu FIN Flood

Štruktúra súborov je uvedená v príručke pre verziu bmIDS-v2.0 [1].

4.2 Konfiguračný súbor

Konfiguračný súbor – config.xml používa aplikácia bmIDSanalyzer, ktorá si z neho pri spustení načíta potrebné údaje pre pripojenie k databázovému serveru, údaje pre pripojenie ku kolektoru a konfiguračné údaje pre samotný analyzér. Štruktúra tohto súboru je jednoduchá a ľahko čitateľná, pretože sú údaje zapísané vo formáte xml. Administrátor má možnosť upraviť tieto údaje podľa potreby. Zoznam konfigurovateľných parametrov s ich popisom je uvedený v tabuľkách tab. 4 – 1 a tab. 4 – 2 .

Tabuľka 4 – 1 Popis parametrov konfiguračného súboru

Parameter	**Popis**
IDS settings	
threshold	prahová hodnota miery podozrenia, ktorej prekročenie signalizuje útok
Database setting - údaje k databáze ktorú používa kolektor	
ip	IP adresa databázoveho servera Postgresql
port	port na ktorom beží databázový server
name	názov databázy s používanymi tabuľkami
login	prihlasovacie meno k databáze
password	prihlasovacie heslo k databáze
ACP setting	
ip	IP adresa servera na ktorom beží kolektor
port	port na ktorom čaká kolektor pre pripojenie analyzéra
user	meno pre pripojenie ku kolektoru
password	heslo pre autentifikáciu ku kolektoru

Tabuľka 4 – 2 Zoznam konfigurovateľných parametrov

Parameter	**Popis**
Mail setting	
sendMail	voliteľné parametre sú *true* a *false*, pri konfigurácii *true* je posielanie emailových správ, v prípade detekovani útoku, zapnuté, pri konfigurácii *false* vypnuté
mailFrom	emailový účet aplikácie bmIDSanalyzer
mailFromPwd	heslo k emailovému účtu aplikácie bmIDSanalyzer
mailTo	emailový účet používateľa, na tento účet budú posielané emailové notifikácie v prípade útoku
SLAweb database setting - údaje k databáze ktorú používa webová aplikácia SLAmeter	
ip	IP adresa databázoveho servera Postgresql
port	port na ktorom beží databázový server
name	názov databázy s používanymi tabuľkami
login	prihlasovacie meno k databáze
password	prihlasovacie heslo k databáze
Standard traffic values	
portScan	štandardné hodnoty charakteristík pre útok skenovanie portov
synFlood	štandardné hodnoty charakteristík pre SYN záplavový útok
udpFlood	štandardné hodnoty charakteristík pre UDP záplavový útok
rstFlood	štandardné hodnoty charakteristík pre RST záplavový útok
ttlFlood	štandardné hodnoty charakteristík pre TTL záplavový útok
finFlood	štandardné hodnoty charakteristík pre FIN záplavový útok

5 Preklad programu

5.1 Zoznam zdrojových textov

Program bmIDSanalyzer:

balík – *sk.tuke.cnl.bmIDS:*

- `Application.java`
- `Config.java`
- `Constants.java`
- `DBclient.java`
- `IPFlow.java`
- `IPFlowReceiver.java`
- `Learn.java`

balík – *sk.tuke.cnl.bmIDS.web:*

- `RedisServer.java`

balík – *sk.tuke.cnl.bmIDS.processor:*

- `TrafficProcessor.java`
- `PortScanProcessor.java`
- `SynFloodProcessor.java`
- `UdpFloodProcessor.java`
- `RstFloodProcessor.java`
- `TTLFloodProcessor.java`
- `FinFloodProcessor.java`

balík – *sk.tuke.cnl.bmIDS.detector:*

- `FuzzyDetector.java`
- `PortScanDetector.java`
- `SynFloodDetector.java`
- `UdpFloodDetector.java`
- `RstFloodDetector.java`
- `TTLFloodDetector.java`
- `FinFloodDetector.java`

balík – *sk.tuke.cnl.bmIDS.traffic:*

- `PortScanSignTraffic.java`
- `SynFloodSignTraffic.java`
- `UdpFloodSignTraffic.java`
- `RstFloodSignTraffic.java`
- `TTLFloodSignTraffic.java`
- `FinFloodSignTraffic.java`

Program bmIDSweb:

balík – *sk.tuke.cnl.bmIDS.client:*

- `Application.java`
- `Config.java`
- `Dbs.java`
- `Filter.java`

- `HomePage.java (.html)`
- `MonitoringPage.java (.html)`
- `AttackLogsPage.java (.html)`
- `SettingsPage.java (.html)`
- `SynFloodDetailPage.java (.html)`
- `UdpFloodDetailPage.java (.html)`
- `PortScanDetailPage.java (.html)`
- `RstFloodDetailPage.java (.html)`
- `TTLFloodDetailPage.java (.html)`
- `FinFloodDetailPage.java (.html)`

balík – *sk.tuke.cnl.bmIDS.client.dataModel:*

- `AttackRecordsProvider.java`
- `AttackRecord.java`
- `AttackDetail.java`

balík – *sk.tuke.cnl.bmIDS.client:*

- `SynFloodCharts.java`
- `UdpFloodCharts.java`
- `PortScanCharts.java`
- `RstFloodCharts.java`
- `TTLFloodCharts.java`
- `FinFloodCharts.java`

5.2 Požiadavky na technické prostriedky pri preklade

Pre zaistenie spoľahlivého behu bmIDS analyzéra sa vyžaduje nasledovná hardvérová konfigurácia:

- CPU Intel Pentium III 1GHz alebo ekvivalent
- RAM 1 GB
- HDD 100 MB voľného priestoru
- sieťová karta 100Mbit/s

Požiadavky na technické prostriedky a samotná inštalácia webovej aplikácie nástroja SLAmeter sú uvedené v príslušnej dokumentácii [4]. Inštalácia exportéra a kolektora sa prevedie tiež podľa príslušnej dokumentácie.

5.3 Požiadavky na programové prostriedky

Pre spustenie webovej aplikácie ids je nutné, aby boli na počítači nainštalované programové prostriedky podľa príslušnej dokumentácie webovej aplikácie nástroja SLAmeter [4]:

Spustenie bmIDS analyzéra vyžaduje:

- operačný systém Linux alebo Windows, ale odporúča sa Linux/Ubuntu
- Java Runtime Enviroment (JRE 6.0)
- databázový server PostgreSQL 7.3 a vyšší
- aplikáciu kolektora JXColl v3.9 podľa jej dokumentácie
- aplikáciu exportéra mybeem podľa jej dokumentácie

5.4 Vlastný preklad

Preklad zdrojových textov je možné vykonať vo vývojovom prostredí Netbeans nasledujúcim spôsobom:

- otvoriť projekt – kliknúť na položku menu File – Open Project a vybrať názov projektu
- spustiť preklad programu – kliknúť pravým tlačidlom na názov projektu a zvoliť položku Build
- spustiteľný jar súbor sa po preložení programu nachádza v adresári dist daného projektu

6 Náväznosť na iné programové produkty

Systém bmIDS získava údaje zo sieťovej prevádzky pomocou exportéra (mybeem) a kolektora (JXColl), a preto závisí na týchto komponentoch, bez ktorých detekciu anomálií nemohol vykonávať. Na prenos údajov od kolektora používa aplikácia bmIDSanalyzer aplikačné rozhranie ACPapi, ktoré je tiež nevyhnutnou podmienkou pre chod systému. Pre poskytovanie vyhodnotených údajov je potrebná webová aplikácia nástroja SLAmeter, v ktorej je implementovaná aplikácie ids, ktorá predstavuje webovú aplikáciu systému bmIDS.

Literatúra

[1] UJLAKY, M.: *Systém bmIDS pre detekciu narušenia v počítačových sieťach*, Diplomová práca, Príloha B, KPI FEI TU, Košice, 2012.

[2] DEMČÁK, D.: *Systém pre detekciu narušenia založený na architektúre IPFIX*, Bakalárska práca, Príloha B, KPI FEI TU, Košice, 2012.

[3] ZÁVADA, V.: *Systém pre detekciu narušenia siete založený na IPFIX*, Bakalárska práca, Príloha B, KPI FEI TU, Košice, 2009.

[4] JUHÁR, J.: *Webová aplikácia nástroja SLAmeter*, Diplomová práca, Príloha B, KPI FEI TU, Košice, 2014.

[5] BERTA, L.: *Systémy pre detekciu narušenia sietí*, Bakalárska práca, Príloha B, KPI FEI TU, Košice, 2013.