UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA: Calculatoare

Monitorizarea unei rețele prin protocolul SNMP

LUCRARE DE LICENȚĂ

Coordonator științific
Ș. l. Ing. Amarandei Cristian Mihai

Absolvent
Ionuț Lumină

Iași, 2016

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII LUCRĂRII DE LICENȚĂ

Subsemnatul(a) LUMINĂ IONUȚ,
legitimat(ă) cu CI seria MZ nr. 083882, CNP 1930609225898
autorul lucrării MONITORIZAREA UNEI REȚELE PRIN PROTOCOLUL SNMP

elaborată în vederea susținerii examenului de finalizare a studiilor de licență organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi" din Iași, sesiunea SEPTEMBRIE a anului universitar 2015-2016, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi" din Iași (Manualul Procedurilor, UTI.POM.02 – Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data 13.09.2016

Semnătura

Cuprins

Monitorizarea unei rețele prin protocolul SNMP

Ionuț Lumină

Rezumat

Proiectul își propune realizarea unui instrument de monitorizare a unei rețele. Soluția propusă se bazează pe protocolul SNMP. Scopul acestui proiect este de a observa, în timp real, starea de funcționare a echipamentelor de comunicație sau a echipamentelor destinate anumitor servicii. Printre lucrurile observabile se enumeră traficul de download și upload, memoria RAM disponibilă, procentul de încărcare al CPU și spațiul liber pe hard disk.

Accesul la aplicație se realizează prin intermediul unei pagini web. Serverul web Apache este pe același echipament pe care este realizat și acest proiect. Prin intermediul paginii web, se pot observa graficele corespunzătoare datelor furnizate de obiectele monitorizate.

Limbajul de programare utilizat, pentru implementarea comunicării dintre obiectul monitorizat și server, este realizat în Python. Datele preluate de aplicație sunt salvate într-o bază de date RRD și apoi sunt prelucrate cu RRDTools.

Pentru partea de interfață cu utilizatorul am utilizat Django și Jinja2 împreună cu Python. Cu ajutorul Jinja2, folosit împreună cu Python. Django a fost folosit pentru a crea două template-uri care au ajutat la crearea mult mai ușoară a paginii web.

Ca o idee generală, am preferat ca aplicația să fie open-source, prin urmare am folosit soluții de tip open-source. De asemenea, sistemul de operare, pe care este realizat proiectul, este CentOS 7.

Capitolul 1. Introducere

Am ales dezvoltarea unei unui instrument de monitorizare a unei rețele. Prin „rețea”, mă refer la rețeaua locală (LAN) a unei organizații.

Această idee a venit în urma unei necesități, la locul de muncă, de a observa, în timp real, starea de funcționare a echipamentelor de comunicație sau a echipamentelor destinate anumitor servicii.

Monitorizarea rețelei pentru o organizație este o funcție critică și deosebit de importantă, care poate salva resurse financiare importante, crește performanța rețelei și costurile de întreținere a infrastructurii. Acest proces presupune colectarea informațiilor referitoare la toate aspectele unei rețele.

Un aspect ce trebuie stabilit, în cazul strategiei de monitorizare are în vedere partea care se dorește a fi observată, care sunt procesele care fac obiectul acestor operații complexe.

Informațiile obținute prin monitorizare pot fi folosite pentru a determina dacă resursele implicate sunt utilizate corespunzător, pentru a verifica cum lucrează echipamentele care sunt folosite în rețea, a urmări activitatea în cadrul unei rețele și pentru a identifica probleme care apar și de a lua măsurile necesare pentru rezolvarea lor.

Monitorizarea rețelei se poate face prin utilizarea diferitelor soluții software sau combinații de echipamente hardware. În cazul acestui proiect am preferat utilizarea soluției software.

Componenta software este formată din scripturi realizate în Python, un fișier de configurare și un fișier de log-uri. Scripturile obțin informațiile de la dispozitivele monitorizate și le salvează în mai multe fișiere cu extensia rrd, apoi generează graficele ce vor fi afișate într-o pagina web, cu ajutorul a două șabloane realizate în Jinja2, dar și cu ajutorul framework-ului Django. În fișierul de configurare sunt informații despre dispozitivele monitorizate și ce anume se dorește a se monitoriza. În fișier-ul de log-uri sunt scrise erorile ce pot apare în funcționarea scripturilor.

Capitolul 2. Fundamentarea teoretică și documentarea bibliografică

2.1. Simple Network Management Protocol

SNMP[1] este un protocol, care funcționează la nivelul aplicație al modelului TCP/IP. Acest protocol a apărut din nevoia companiilor de a evita situațiile neplăcute în care componentele unei rețele să nu funcționeze la parametri optimi.
Majoritatea implementărilor folosesc UDP pentru transferul de mesaje, deoarece este considerat acceptabilă pierderea de pachete în comparație cu funcțiile pe care trebuie să le îndeplinească entitățile administrate.
UDP (User Datagram Protocol) este un protocol fără conexiune, între emitor și receptor și nu oferă siguranța livrării mesajelor, acestea fiind responsabilitatea celor care a implementat aplicația.

SNMP se bazează pe modelul manager / agent, care constă într-un administrator, un agent și o bază de date de management a informațiilor , gestionată de obiecte de protocol și de rețea.

Managerul oferă interfața dintr om și sistemul de management.

Agentul oferă interfața între manager și dispozitivele fizice ce urmează a fi gestionate, cum ar fi bridge-uri, hub-uri, routere, servere de rețea sau computer-ul unui utilizator. Aceste obiecte gestionate ar putea fi hardware, parametrii de configurare, statisticile de performanță și altele.

Aceste obiecte sunt aranjate în ceea ce este cunoscut ca bază de date a informațiilor virtuale, denumită Baza de gestionare a informațiilor, de asemenea, numită și MIB (Management Information Base). SNMP permite managerilor și agenților de a comunica cu scopul de a accesa aceste obiecte.

Managerul și agentul utilizează baza de gestionare a informațiilor și un set de comenzi relativ mic, pentru a schimba informații. MIB este organizat într-o structură de arbore cu variabile individuale, cum ar fi punctul de stare sau descriere, fiind reprezentat ca frunzele de pe ramuri (vezi Figura 2.1 [2]). Un tag numeric lung sau un obiect de identificare (OID) este folosit pentru a distinge fiecare variabilă unică în MIB și în mesajele SNMP.

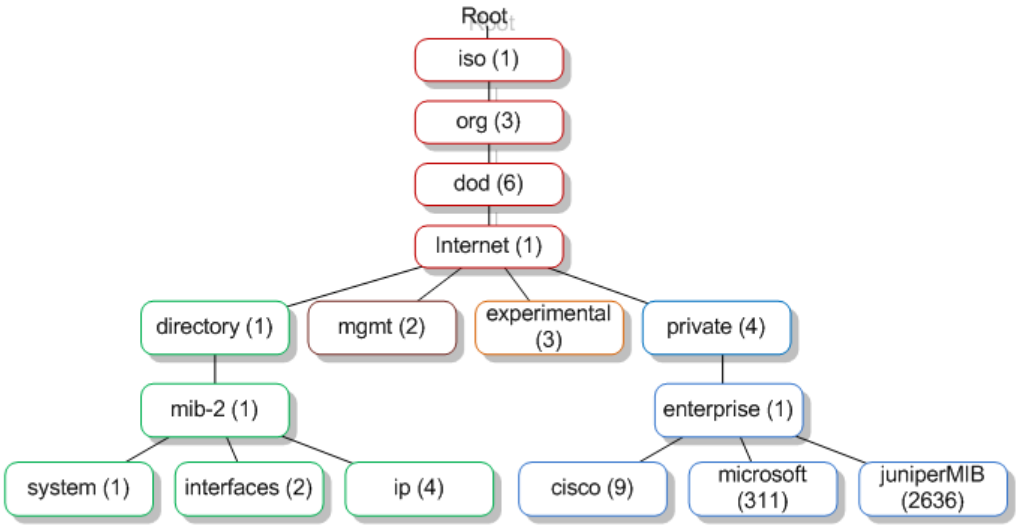

Figura 2.1: Exemplu de arbore OID

Managerul SNMP nu poate monitoriza dispozitive cu excepția cazului în care le-a compilat fișierele MIB. MIB-ul este, de asemenea, un ghid de capabilități a dispozitivelor SNMP.

Fiecare elemente SNMP gestionează obiecte specifice cu fiecare obiect având caracteristici specifice. Fiecare obiect/caracteristică are un obiect unic de identificare constând din numere separate prin puncte zecimale (de exemplu: 1.3.6.1.4.1.2636.1).

Unele protocoale sunt orientate pe octet, pe când altele sunt orientate pe pachete. Pachetele conțin antet, date și octeți de control. Protocolul SNMP este orientat pe pachete (vezi Figura 2.2 [3]), PDU (protocol Data Unit).

Un PDU are următoarele câmpuri:

- Version – versiunea de SNMP;
- Community – denumirea grupului din care face parte;
- Request ID – este trimis înapoi la manager împreună cu răspunsul pentru a ști la ce cerere se referă un răspuns;
- Error code – agentul plasează un cod de eroare în acest câmp, dacă are loc o eroare la procesarea cererii;
- OID – obiectul din MIB;
- Value – valoarea care se dorește setată, dacă mesajul este un SetRequest, o valoare nulă dacă mesajul este un GetRequest și valoarea returnată pentru OID, dacă mesajul este un GetResponse.

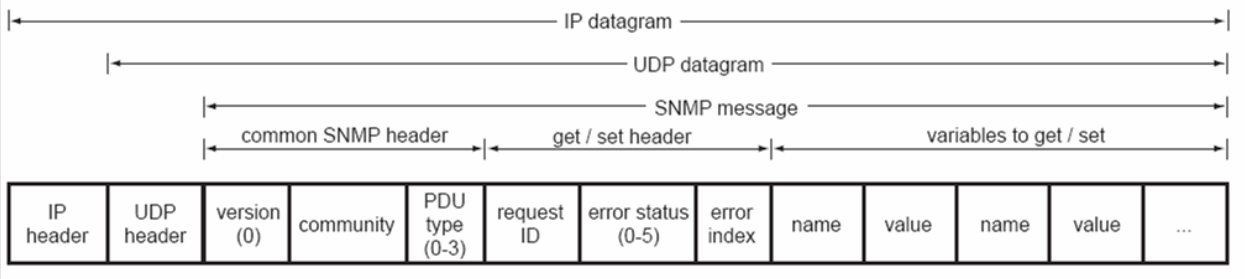

Figura 2.2: Formatul unui PDU

Există 7 tipuri de PDU:

- GetRequest – o cerere manager-agent pentru a primi o valoare sau o listă de valori a unui obiect din MIB;
- SetRequest – o cerere de la manager la agent pentru a seta sau modifica o valoare;
- GetNextRequest – managerul solicită valoarea obiectului care urmează după obiectul precizat în cerere;
- GetBulkRequest – o versiune optimizată a GetNextRequest, managerul poate cere mai multe iterații de GetNextRequest;
- GetResponse – mesajul trimis de agent, ca urmare a cererii de tip GetRequest, GetNextRequest sau SetRequest;
- Trap – o atenționare trimisă de agent către manager, atunci când apare un eveniment neașteptat legat de dispozitivul monitorizat;
- InformRequest – asigură livrarea mesajelor Trap.

De-a lungul timpului au fost dezvoltate mai multe versiuni al SNMP.

2.1.1. SNMPv1

SNMP [4] versiunea 1 este versiunea inițială a conceptului de SNMP, a fost introdusă pentru a răspunde nevoii de administrare a dispozitivelor ce folosesc protocolul IP. Este definit în RFC 1157.

SNMP oferă utilizatorilor un set simplu de operații, care permite acestor dispozitive să fie administrate de la distanță. Baza protocolului este un set de operații, care oferă administratorilor posibilitatea de a urmări sau modifica parametrii unor dispozitive ce suportă SNMP.

Prima versiunea a fost criticată pentru lipsa de securitate. Autentificarea se făcea printr-un grup numit *community string* și era nu era codificată sub nicio formă.

2.1.2. SNMPv2

SNMPv2 [4], revizuiește prima versiune și aduce îmbunătățiri la performanță, securitate, confidențialitate și comunicarea între manageri. Pe lângă acestea, versiunea 2 include *GetBulkRequest* și detalierea mesajelor de eroare raportate către manager. *GetBulkRequest* suportă aducerea de tabele și cantități mari de date.

SNMPv2c (Community-Based Simple Network Management Protocol version 2) este definită în RFC 1901 – RFC 1908. Această versiune a fost considerată, până să apară versiunea a treia, un standard pentru SNMP.

2.1.3. SNMPv3

Versiunea a treia [4] nu aduce schimbări, în afară de partea de securitate. Cu toate acestea conține termeni diferiți față de versiunile anterioare.

Această versiune are o structură modulară, permițând adăugări ușoare, dar și modificări.

Modificările pe partea de securitate se referă la autentificare, criptare și controlul accesului. Autentificarea permite doar surselor autorizate să genereze cereri SNMP. Acest lucru se realizează prin crearea de cont de utilizator și respectiv a unei parole. Criptarea previne citirea sau modificarea mesajelor SNMP transmise prin rețea, iar controlul accesului la MIB-uri a fost implementat pentru a limita accesul la anumite informații.

Din punct de vedere al termenilor, versiunea a treia nu folosește termeni de agent și manager, ci de entități. Fiecare entitate având un Engine și un modul software de funcții ce inițiază sau răspund la cereri SNMP. Engine-ul furnizează securitatea, controlul de acces și procesarea mesajelor.

Versiunea a treia este compatibilă cu versiunile anterioare, lucru pe care nu-l putem spune și față de versiunea a doua.

2.2. RRDtool

RRDtool[5] (Round-Robin Database tool) poate prelucra date precum lățimea de bandă a unei rețele, temperatura CPU, procentul de încărcare a CPU, memoria RAM utilizată, spațiul de stocare utilizat și altele. Aceste date sunt stocate într-o bază de date cu buffer circular.

RRDtool a fost conceput inițiat pentru a extrage date RRD într-un format grafic.

El poate fi folosit în limbaje precum Perl, Python, Ruby, Tcl, PHP și Lua. De asemenea există și o versiunea independentă realizată în Java.

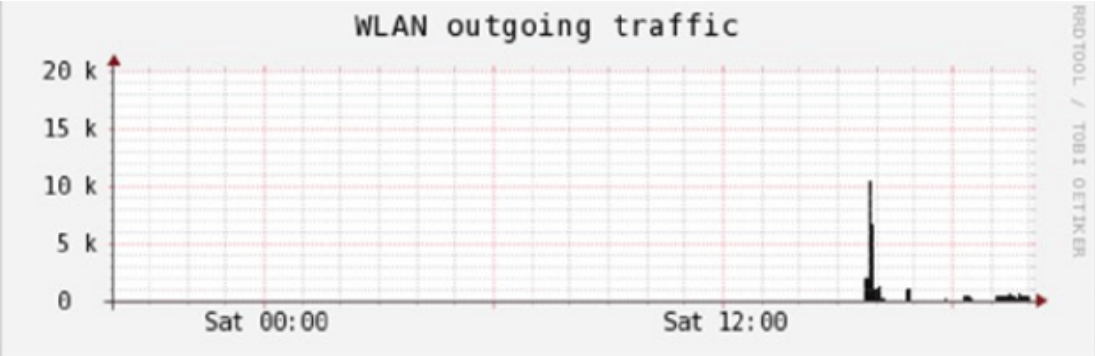

Figura 2.3: Exemplu de grafic RRD

2.3. Limbajul Python și avantajele folosirii acestuia

Limbajul[6] de programare Python a fost creat de Guido van Rossum, în anul 1989. Descendent al limbajului de programare ABC.

Python este un limbaj de programare dinamic, de nivel înalt, ce pune accent pe expresivitatea și înțelegerea ușoară a codului. Sintaxa sa permite implementări echivalente cu alte limbaje în mai puține linii de cod. Datorită acestui fapt, Python este foarte răspândit atât în programarea de aplicații, cât și în zona de scripting.

Spre deosebire de C, Python nu este un limbaj compilat, ci interpretat. Acest fapt are atât avantaje, cât și dezavantaje. Pe de-o parte, Python este mai lent decât C. Pe de altă parte, aplicațiile Python sunt foarte ușor de depanat, codul putând fi ușor inspectat în timpul rulării. De asemenea, este foarte ușor de experimentat cu mici fragmente de cod folosind interpretorul Python.

Sintaxa este gândita în așa fel încât programele Python să fie ușor de citit. Acest lucru este obținut prin folosirea de cuvinte în locul semnelor (de exemplu, *and* în loc de &&) și prin includerea indentării în limbaj. Astfel, în Python nu se folosesc acolade (ca în C/C++, Java), ci blocurile de cod se delimitează prin indentare. Programele Python sunt, de multe ori, foarte aproape de o implementare echivalentă în pseudo cod.

Includerea diferitelor structuri și funcții ce permit manipularea și prelucrarea lor, precum și multe biblioteci de funcții sunt prezente datorită conceptului “Batteries Included”. Guido van Rossum și comunitatea, ce s-a format în jurul limbajului, cred că un limbaj de programare nu prezintă utilitate practică dacă nu are un set de biblioteci importante pentru majoritatea dezvoltatorilor.

Din acest motiv Python include biblioteci pentru lucrul cu fișiere, arhive, fișiere XML și un set de biblioteci pentru lucrul cu rețeaua și principalele protocoale de comunicare pe internet (HTTP, Telnet, FTP). Un număr mare de platforme Web sunt construite cu Python. Abilitățile limbajului ca limbaj pentru programarea CGI sunt în afara oricăror dubii. De exemplu YouTube, unul din site-urile cu cea mai amplă cantitate de trafic din lume, este construit pe baza limbajului Python.

Totuși, Python permite extinderea funcționalității prin pachete adiționale programate de terți care sunt axate pe o anumită funcționalitate. De pildă, pachetul *wxPython* conține metodele și structurile necesare creării unei interfețe grafice.

Popularitatea limbajului este în creștere începând cu anul 2000, datorită faptului că Python permite crearea mai rapidă a aplicațiilor, care nu cer viteze înalte de procesare a datelor. De asemenea este util ca limbaj de scripting, utilizat în cadrul aplicațiilor scrise în alte limbaje.

Modulele (bibliotecile) Python pot fi de asemenea scrise în *C*, compilate și importate în Python pentru a mări viteza de procesare.

2.4. Framework-ul Django

Django [7] este un soft pentru dezvoltarea aplicațiilor web, open-source, realizat în Python. El urmează modelul arhitectural Model-View-Controller.

Acest framework a fost creat de Adrian Holovay și Simon Willison, în anul 2003. El a fost realizat ca și răspuns la necesitatea de a publica cât mai repede articole (creatorii acestui framework, lucrau la ziarul Lawrence Journal-World).

Scopul principal al acestui soft, pentru dezvoltarea aplicațiilor web, este de a facilita crearea de website-uri complexe, ce au în componența lor și o bază de date. Django pune accent pe reutilizarea codului, pe modularitate, dezvoltare rapidă a site-urilor web, ghidându-se după principiul DRY („Don`t repeat yourself”). Django este realizat în Python, chiar și fișierele de configurare și modelele de date sunt implementate în acest limbaj de programare. Django oferă și un panou administrativ, prin intermediul căruia se pot crea, citi, actualiza și șterge date din baza de date.

Printre site-urile care utilizează Django se enumeră Pinterest, Instagram, Mozilla și The Washington Times.

2.5. Jinja2

Template-engine-ul Jinja2 [8] permite utilizarea codului Python pentru realizarea site-urilor web.

Jinja2 este un limbaj text ce îți permite să generezi orice marcaj precum și cod sursă. Jinja2 este asemenea lui Django, cu excepția că în Jinja2 ai posibilitatea să apelezi funcții cu argumente. Template-ul Jinja2 oferă posibilitatea personalizării codului prin tag-uri și filtre.

2.6. Serverul web Apache

Apache [9] este un server HTTP de tip open-source. Apache este folosit, în prezent, de aproximativ 65.2 % din paginile web. El a reprezentat o alternativă viabilă la Netscape Communications Corporation și a evoluat rapid în funcționalitate și performanță ca un rival competitiv pentru alte servere web bazate pe Unix.

Apache este dezvoltat de o comunitate deschisă, sub emblema Apache Software Foundation. Aplicația este disponibilă pentru o mare varietate de sisteme de operare incluzând Unix, FreeBSD, Linux, Solaris, Mac OS și Microsoft Windows.

Serverul Apache este caracterizat ca fiind un software gratuit și open-source, aceasta făcând ca, începând din aprilie 1996, să devină cel mai popular web server.

Prima versiune a fost creată de Robert McCool, care la vremea respectivă era implicat în proiectul Național Center for Supercomputing Applications (NCSA HTTPd). A doua versiune a serverului a fost o rescriere substanțială, punându-se accent pe crearea unui layer prioritar și a suportului de module.

Apache suportă o mare varietate de module care îi extind funcționalitatea. Aceasta variază de la server side programming, până la scheme de autentificare. Câteva limbaje suportate sunt Pearl, Python, Tcl, PHP. O altă calitate a serverului Apache este găzduirea virtuală, care constă în posibilitatea de a găzdui mai multe site-uri simultan pe același server.

Google folosește o versiune modificată de Apache, numită Google Web Server. De asemenea și proiectele Wikimedia rulează pe un server Apache.

2.7. Soluții existente pentru monitorizarea unei rețele

Pentru monitorizare se pot folosi multe aplicații software, unele dintre ele sunt gratis, altele trebuie cumpărate Printre acestea putem enumera Nagios, Cacti sau Zenoss.

Fiecare dintre acestea au fost realizate cu ajutorul diferitelor limbaje de programare, dar toate au în comun RRDtool și un server web.

Proiectul prezentat îmbină limbajul de programare Python cu framework-ul Django și template-ul Jinja2, lucru pe care nu l-am văzut la alt produs pentru monitorizarea unei rețele.

În cele ce urmează voi prezenta câteva produse existente cu ajutorul cărora poți monitoriza o rețea.

2.7.1. Nagios

Nagios [10] este un sistem open-source de monitorizare a sistemelor informatice și a rețelelor de calculatoare, realizat cu ajutorul limbajului de programare C. Sistem de monitorizare a nodurilor sau a serviciilor de rețea și în cazul în care apar probleme, notifică administratorul. Monitorizarea continuă a tuturor componentelor dezvăluie zonele cu probleme și ajută la rezolvarea lor înainte de eșec, fără a afecta performanța rețelei. Nagios monitorizează activitatea serviciilor de rețea precum: SMTP, POP3, IMAP, SSH, Telnet, FTP, HTTP, DNS și multe altele. De asemenea, el poate fi folosit pentru a monitoriza utilizarea resurselor de pe server, precum: procentul de încărcare a CPU, memoria RAM utilizată, spațiul de stocare utilizat și altele. Nagios nu funcționează doar în Unix, ci și în distribuții Linux, dar și în sisteme de operare precum Mac OS și Windows.

Cu ajutorul Nagios poți monitoriza un sistem de la distanță. Arhitectura simplă poate crea propriile căi de servicii de testare și manipulări de evenimente (spre exemplu, repornirea unui serviciu de rețea).

Nagios este apreciat pentru abilitatea de a construi hărți ale infrastructurii de rețea și grafice a parametrilor diferitelor sisteme care sunt monitorizate.

Proiectul are originea încă din 2002, deși la început era cunoscut sub numele de NetSaint. Principalul dezvoltator al acestui produs este Ethan Galstat. Față de versiunea inițială proiectul a evoluat, permițând utilizarea plugin-urilor și a addon-urilor.

2.7.2. Cacti

Cacti [11] este un alt sistem open-source pentru monitorizarea sistemelor informatice și a rețelelor de calculatoare. Acesta este o aplicație front-end realizată în PHP.

Cacti a fost lansat în anul 2001, de către Ian Berry. Acest proiect a fost realizat în urma necesității de a monitoriza rețeaua unui campus. El învățând în timpul liceului PHP și MySQL, a reușit să le îmbine, realizând Cacti. Principalul lui scop în realizarea acestui proiect a fost pentru a oferi o alternativă mai ușor de folosit decât RRDtool și mai flexibilă decât MRTG.

Ca și utilizare comună este folosit pentru a monitoriza traficul sau încărcarea CPU a unor servere sau a unor switch-uri sau routere. Toate acestea sunt realizate prin intermediul SNMP.

În prezent, Cacti oferă două versiuni, una realizată în PHP, ce este recomandată rețelelor de mici dimensiuni, cea de a doua este realizată în C și este recomandată pentru monitorizarea rețelelor la o scară mai mare.

2.7.3. Zenoss

Zenoss [12] este o aplicație open-source, parțial gratuită. Dezvoltarea aplicației a început în anul 2002 de Erik Dahl, iar 4 ani mai târziu a fost lansată prima versiune.

Zenoss a fost realizată cu ajutorul limbajelor de programare Python și Java.

Această aplicație oferă o interfață web cu ajutorul căreia poți monitoriza activitatea, configurările și anumite evenimente dintr-o rețea.

Versiunea plătita a acestui produs oferă pe lângă suport, integrarea aplicației cu Microsoft SQL și Exchange.

În prezent, Zenoss rulează doar pe distribuțiile Fedora (Red Hat Enterprise Linux și CentOS).

Capitolul 3. Proiectarea aplicației

3.1. Componenta hardware

După cum spuneam și în introducere, proiectul a venit ca răspuns a necesității la locul de muncă pentru a monitoriza diferite stații de lucru. În cele ce urmează voi prezenta obiectele care au ajutat la funcționarea acestui proiect (vezi Figura 3.1 pentru o vizualizarea mai concretă a infrastructurii).

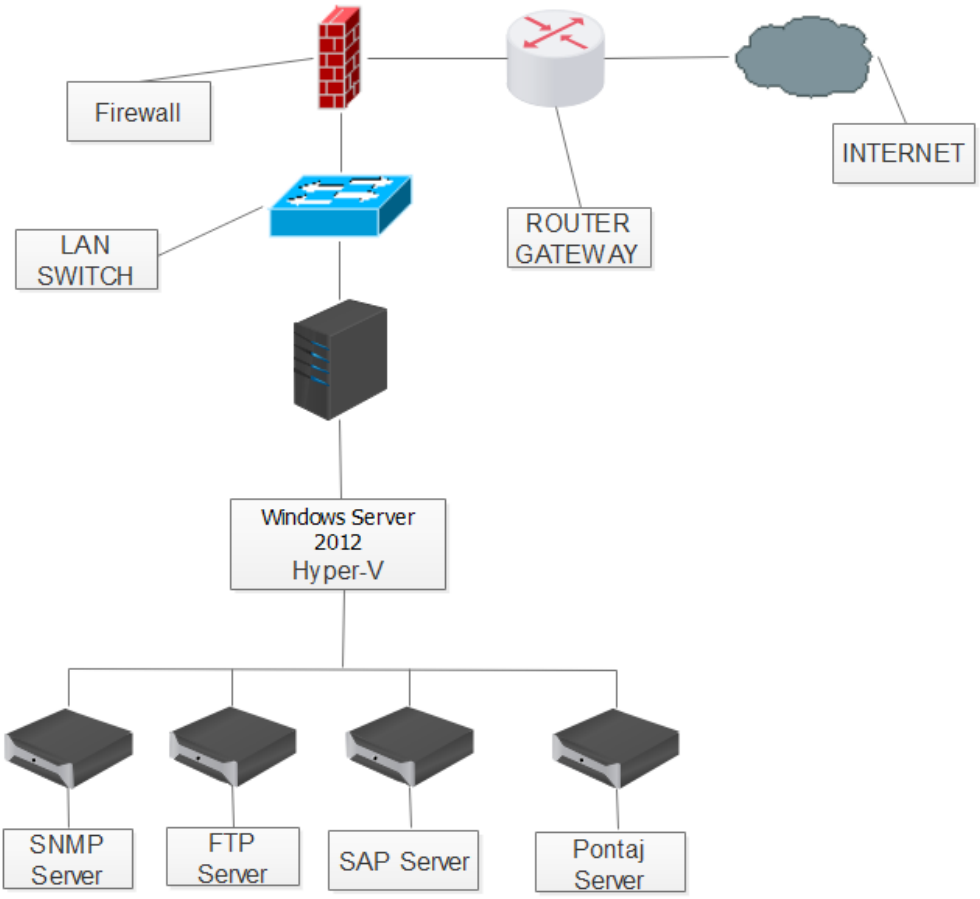

Figura 3.1: Infrastructura hardware a companiei

3.1.1. Router

Router-ul [13] este un dispozitiv hardware sau software, care conectează două sau mai multe rețele de calculatoare, bazate pe comutare de pachete. Funcția îndeplinită de routere se numește routare.

Routerele operează la nivelul 3 al modelului OSI, Ele folosesc adrese IP (de rețea) ale pachetelor aflate în tranzit pentru a decide către care interfață de ieșire să trimită pachetul respectiv. Decizia este luată comparând adresa calculatorului destinație cu înregistrările din tabela de routere. Aceasta poate conține atât înregistrări statistice (configurate/definite de către administratorul rețelui), cât și dinamice, aflate de la routerele vecine prin intermediul unor protocoale de rutare.

În cazul de față, routerul este un dispozitiv hardware și este folosit pentru interconectarea

între providerul de internet (mai exact între două sisteme autonome) și rețeaua companiei din care fac parte.

3.1.2. Firewall

În rețelele de calculatoare, un firewall [13], denumit și paravan de protecție este un dispozitiv sau o serie de dispozitive configurate în așa fel încât să filtreze, să cripteze sau să intermedieze traficul între diferite domenii de securitate pe baza unor reguli predefinite.

Un firewall este un dispozitiv hardware sau software, care monitorizează și filtrează permanent transmisiile de date realizate între rețeaua locală și internet.

Soluțiile de protecție prin firewall se împart în două categorii:

- soluții profesionale hardware sau software dedicate protecției întregului trafic dintre rețeaua unei companii sau instituții și internet;
- firewall dedicat monitorizării traficului pe calculatorul personal.

În acest proiect firewall-ul este un dispozitiv hardware și are rolul de a filtra traficul dintre rețeaua companiei și internet.

3.1.3. Switch

Un switch [13], numit și comutator de rețea este un dispozitiv care realizează interconectarea diferitelor segmente de rețea, pe baza adreselor MAC. Dispozitivele hardware uzuale includ switch-uri, care utilizează conexiuni de 10, 100 sau chiar 1000 MB pe secundă, la semi-duplex sau duplex.

Semi-duplex înseamnă că dispozitivul poate doar să trimită sau să primească informații, la un moment dat, în timp ce duplex oferă posibilitatea trimiterii și primirii concomitente de informații.

În cazul de față switch interconectează firewall-ul, calculatoarele angajaților și serverul pe care este realizat acest proiect.

3.1.4. Server

Un server [13] este o aplicație pe un calculator sau chiar un calculator întreg, care operează continuu în rețeaua din care face parte și așteaptă solicitări din partea altor calculatoare din rețea. Serverul poate fi folosit simultan și pentru alte scopuri, dar poate îndeplini exclusiv funcția de server. De exemplu, un calculator se poate folosi într-o companiei, simultan pentru două scopuri, ca stație de lucru și ca server pentru celelalte calculatoare din birou.

În prezent, serverele seamănă fizic cu celelalte calculatoare uzuale, deși configurația hardware este deseori optimizată pentru funcționarea lor ca servere. Multe componente hardware sunt identice cu cele pe care le găsim într-un calculator personal. Totuși serverele rulează sisteme de operare și programe specializate, care sunt foarte diferite față de cele folosite pe calculatoarele personale și stațiile de lucru.

Serverele deservesc recursele hardware, care sunt partajate și pot uneori fi comandate de către calculatoarele client, cum ar fi imprimante sau sisteme de fișiere. Această partajare permite un acces și o securitate mai bună. De asemenea, ea poate reduce cheltuielile pentru dispozitive periferice.

În cadrul acestui proiect, serverul are rolul de a reduce cheltuielile pentru dispozitivele periferice. În componența lui se găsesc mai multe servere virtuale, dar aceste vor fi detaliate la componenta software.

3.2. Componenta software

După cum am specificat anterior, în acest context principalul rol al serverului este de a reduce cheltuielile pentru achiziționarea altor dispozitive. Prin urmare, a fost nevoie de instalarea unui mediu virtual.

Principalele medii virtuale de pe piața IT sunt Hyper-V și VMware ESXI. În cazul se față s-a preferat folosirea mediului virtual Hyper-V.

Prin urmare, a fost nevoie de un sistem de operare creat de Microsoft, mai exact Microsoft Windows Server 2012 R2.

3.2.1. Microsoft Windows Server 2012 R2

Microsoft Windows Server 2012 [14] reprezintă a șasea ediție Windows Server. Ea este ediția pentru sisteme de tip server și este succesorul a Windows Server 2008 R2. Două versiuni au apărut în ediția dezvoltării și anume, previzualizare pentru dezvoltatorii de aplicații și versiunea beta. Produsul a devenit disponibil pe piață. Începând cu data de 4 septembrie 2012.

Spre deosebire de versiunea precedentă, Windows Server 2012 oferă patru ediții: Foundation, Essential, Standard și Datacenter.

Windows Server 2012 Foundation este disponibilă doar cu achiziționarea unui nou server. Proiectat pentru întreprinderi mici, această ediție este limitată la 15 utilizatori. De asemenea, această versiune nu suportă virtualizare și suportă doar un singur procesor. În rest, oferă o infrastructură de rețea de bază – Active Directory (AD), acces la distanță, partajare de fișiere și imprimare.

Windows Server 2012 Essential este conceput pentru mediile de afaceri mici (IMM-uri – Întreprinderi Mici și Mijlocii). Această versiune este limitată la 25 de utilizatori. La fel ca și versiunea Foundation, nu suportă virtualizarea, dar suportă două procesoare.

Windows Server 2012 Standard dispune de toate caracteristicile produsului. Această versiune suportă până la două procesoare per licență și suportă virtualizarea.

Windows Server 2012 Datacenter este destinat pentru cei care au utilizarea pe scară largă a mașinilor virtuale. Fiecare licență acoperă până la două procesoare și un număr nelimitat de mașini virtuale. Această versiune suportă mașini cu un maximum de 640 procesoare și 4 TB memorie RAM.

Fiind de la sine înțeles, pe server s-a preferat instalarea Microsoft Windows Server 2012 Datacenter.

Cerințele minime pentru instalarea acestui sistem de operare sunt: o arhitectură pe 64 biți, un procesor cu o frecvență de 1.4 GHz, 512 MB memorie RAM și 32GB spațiul minim pe HDD.

Microsoft Windows Server 2012 R2 este a doua ediție revizuită a Microsoft Windows Server 2012.

3.2.2. Microsoft Hyper-V

Microsoft Hyper-V [15] cunoscut și ca Windows Server Virtualization este un soft nativ folosit la virtualizare.

Un server ce rulează Hyper-V poate fi accesat de la distanță de multiple calculatoare client. Fiecare din calculatoarele client pot rula ca și când ar folosi serverul gazdă, în mod direct. Utilizatorii calculatoarelor client pot rula aplicații de pe serverul gazdă de la distanță, chiar dacă aplicațiile respective nu sunt instalate și pe calculatoarele client.

Microsoft Hyper-V există în două variante:

- Ca produs de sine stătător, numit Hyper-V server – ultima versiune este Hyper-V Server 2012 R2;

- Ca rol instalabil al sistemului de operare pentru Windows Server.

Versiunea de sine stătătoare a Hyper-V, Hyper-V Server include întreaga funcționalitate a Hyper-V. Celelalte roluri ale Windows Server sunt dezactivate, iar serviciile oferite sunt limitate. De asemenea această versiune este limitată la o interfață de tip linie de comandă (CLI), iar configurare sistemului de operare, a componentelor și a software-ului se efectuează cu ajutorul comenzilor din linia de comandă.
Administrarea și configurarea Hyper-V Server, cât și a configurarea sistemelor de operare virtuale este, în general, realizată prin descărcarea consolei extinse, Microsoft Management Consoles ce este instalată pe un calculator client.

Hyper-V se găsește și ca rol instalabil în Windows Server. Această modalitate permite ca configurarea și administrarea Hyper-V să fie mult mai ușoară, datorită interfeței grafice.

3.2.3. Linux

Pentru instalarea serverului pentru monitorizare, am optat pentru o distribuție Linux.

Inițial, Linux [16] era un sistem de operare, care a fost creat ca un hobby de către un tânăr student, Linus Torvalds, la Universitatea din Helsinki, Finlanda. Linus a avut un interes în Minix, un mic sistem de operare derivat din UNIX, și a decis să dezvolte un sistem care să depășească standardele Minix. Și-a început activitatea în 1991, când a lansat versiunea 0.02 și a lucrat în mod constant până în 1994, când versiunea 1.0 a kernel-ului Linux. Kernelul dezvoltat de Linus este inima tuturor sistemelor Linux, fiind dezvoltat și distribuit sub GNU General Public License, iar codul său sursă este disponibil gratuit pentru oricine.

Sistemele de operare bazate pe Linux sunt disponibile, în general, sub formă de „distribuții”. Unele dintre acestea sunt orientate spre utilizatorul particular, pe când altele sunt orientate către servere. Câteva din cele mai folosite distribuții Linux sunt: Ubuntu, SUSE Linux, Fedora, Debian și Gentoo Linux.

Dintre acestea am preferat să folosesc o distribuție bazată pe RedHat Enterprise Linux, mai exact sistemul de operare CentOS.

3.2.4. CentOS 7

Community Enterprise Operating System (CentOS) [17] este un sistem de operare GNU/Linux, open-source, gratuit. Sursele acestui sistem de operare sunt puse la dispoziție sub licență open-source de către Red Hat Enterprise Linux.

Acest sistem de operare a luat naștere pentru a oferi o distribuție gratuită pentru mediul de afaceri.

Versiunea plătită a acestui sistem de operare este Red Hat Enterprise Linux. Diferența majoră între aceste două sisteme de operare este că Red Hat Enterprise Linux oferă suport tehnic.

Fiecare versiune de CentOS este întreținută pentru o perioadă de 7 ani (pentru actualizările referitoare la probleme de securitate). Versiunile noi apar la fiecare 2 ani și sunt actualizate cu regularitate (aproximativ, la fiecare 6 luni), pentru a include suport pentru hardware nou.

În prezent CentOS se află la cea de a șaptea versiune.

Am ales acest sistem de operare, deoarece sunt familiar cu acesta și l-am mai folosit și la alte proiecte.

3.2.5. Aplicația propriu zisă

Aplicația propriu zisă conține mai multe fișiere, fiecare având rolul lui, cu ajutorul cărora funcționează acest proiect.

Primul fișier prezentat este un fișier de configurare. Acest este numit *snmp-manager.cfg*. Acest fișier conține datele obiectelor monitorizate. Un exemplu concret al acestor informații se regăsește în Anexa 1. În acest exemplu se regăsesc informațiile necesare pentru monitorizarea unui server FTP. Informațiile sunt structurate în informații despre obiectul monitorizat și ce informații trimite agentul către manager.

Informațiile ce se găsesc în *snmp-manager.cfg* sunt:

- description – o etichetă care va apărea și pe pagina web, care să specifice ce obiect este monitorizat și ce informație transmite agentul către manager;
- address – adresa IP a obiectului monitorizat;
- port – portul prin intermediul căruia comunică managerul cu agentul;
- communityro – community-ul string necesar pentru securitatea SNMP;
- OID – cu ajutorul acestuia agentul știe ce informații să trimită managerului;
- system – conține eticheta a ce sistem este monitorizat;
- sampling_rate – perioada în secunde, cu care se trimit informațiile de la agent la manager.

Un alt fișier ce ajută la funcționarea acestui proiect este *snmp-manager.py*. Acest fișier conține un script realizat în Python. Cu ajutorul acestui script se culeg informațiile trimise de agenți către manager. Acest script ia informațiile necesare pentru un obiect monitorizat din fișierul *snmp-manager.cfg* le prelucrează pentru a lua datele trimise de agenți către manager și le salvează într-o bază de date RRD; câte o bază de date pentru fiecare tip de informație comunicată de agent către manager.

Pentru prelucrarea bazelor de date, a fost necesară crearea unui alt script, care se găsește în fișierul *snmp-pages.py*. Acest script prelucrează bazele de date transformându-le în fișiere png. De asemenea acest script generează și paginile web ce pot fi observate, cu ușurință, obiectele monitorizate.

Acest script a fost realizat în Python cu ajutorul Jinja2.

Alte două fișiere sunt *template.html* și *details.html*. Aceste două fișiere sunt realizate în Django. Ele ajuta la prelucrarea informațiilor pentru a fi afișate pe pagina web.

Capitolul 4. Implementarea aplicației

Prima cunoștință ce este necesară pentru implementarea aplicației este cum comunică managerul și agentul SNMP.

4.1. Comunicarea manager – agent

Vom continua să examinăm protocolul SNMP, concentrându-ne atenția la modelul stratificat de comunicare, utilizat pentru a face schimb de informații. Această secțiune se concentrează asupra structurii mesajelor SNMP, cu toate acestea un mesaj SNMP nu este trimis de la sine. Acesta este împachetat într-un pachet UDP, care la rândul lui este inclus într-un pachet IP. Acestea sunt cele la care se face referire ca niveluri și se bazează pe un model cu patru niveluri. SNMP este un protocol caracteristic nivelului Aplicație, UDP face parte din nivelul Transport și protocolul IP este caracteristic nivelului Internet (aici ne referim la stiva TCP/IP). Al patrulea nivel este nivelul Acces rețea, unde sunt asamblate pachetele. Acest model mulți-nivel izolează sarcinile de comunicație și în cele din urmă, ajută la elaborarea și punerea în aplicare a unei rețele.

Pentru a ilustra funcționarea acestui model pe niveluri, voi prezenta o solicitare Get, din perspectiva agentului.

Managerul vrea să știe numele sistemului agentului și pregătește un mesaj Get, cu OID-ul corespunzător. Se trece mesajul la nivelul Transport – UDP. Nivelul transport adaugă un bloc de date, care identifică portul managerului la care ar trebui să fie trimis pachetul de răspuns și portul de la care se așteaptă agentul SNMP să asculte mesaje. Pachetele astfel formate sunt trecute la nivelul Internet – IP. Aici, un bloc de date, care conține adresele IP și MAC ale managerului și ale agentului, este adăugat înainte de pachetul întreg asamblat și este trecut la nivelul Acces rețea. Nivelul Acces rețea verifică accesul și disponibilitatea la mediul de comunicație și plasează pachetele pentru transport.

După ce trece de router, pe baza adresei IP, pachetul ajunge, în final, la agent. Aici se trece prin aceleași patru niveluri în exact ordinea opusă, care a fost realizată la manager. În primul rând, este tras din mediul de comunicație interfața de către nivelul de Acces rețea. După ce confirmă că pachetul este intact și valid, nivelul de acces la rețea, pur și simplu trece pachetul la nivelul Internet. La nivelul internet se verifică MAC-ul și adresa IP și îl plasează mai departe nivelului Transport. Aici este verificat portul destinație pentru aplicațiile conectate. Dacă o aplicație ascultă la portul destinație, pachetul este trecut la nivelul Aplicație. În cazul în care aplicația, care ascultă este agentul SNMP, solicitarea GET este procesată. Răspunsul agentului urmează, apoi aceeași cale, dar în sens invers pentru a ajunge la manager. Toate acest lucruri sunt explicate în Figura 4.1[18].

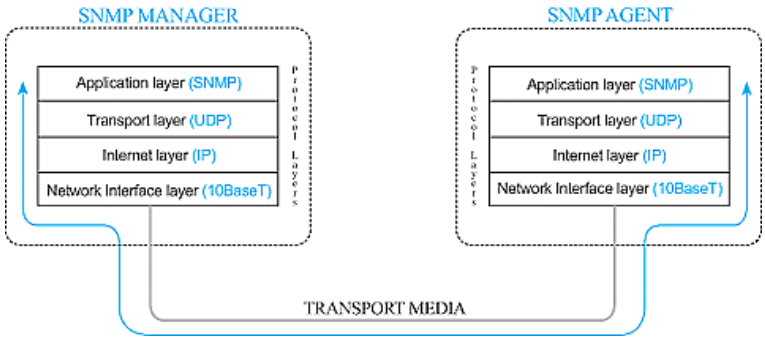

Figura 4.1: Comunicarea între managerul și agentul SNMP

4.2. Interfața cu utilizatorul

La deschiderea paginii web sunt prezentate informații generale despre obiectele monitorizate. Informații precum adresa IP, portul și informațiile ce sunt puse la dispoziție prin SNMP. Adresa la care se găsește interfața cu utilizatorul este asemenea cu adresa IP a serverului SNMP. O prezentare concretă a interfeței se găsește în Figura 4.2.

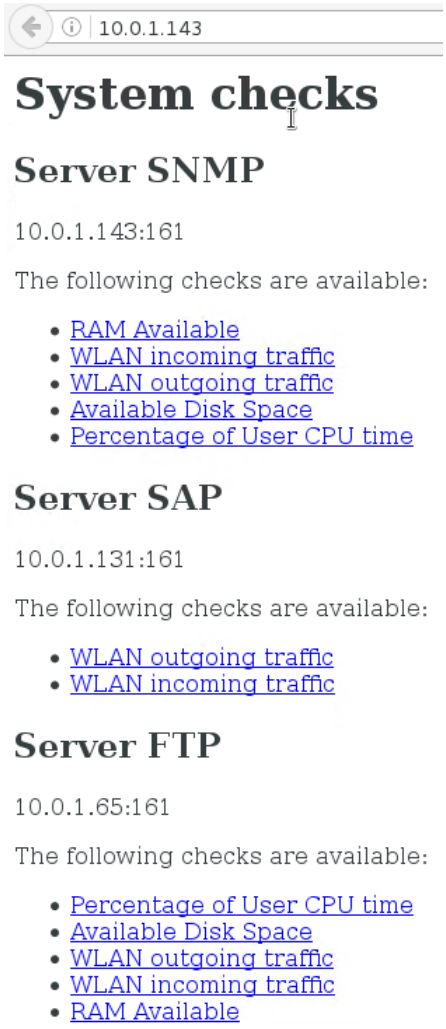

Figura 4.2: Pagina principală a aplicației

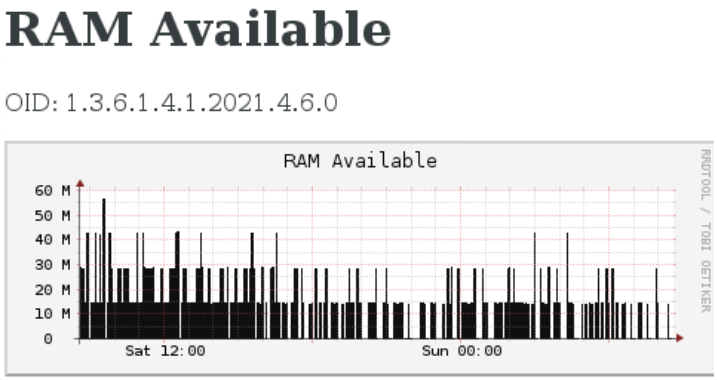

Figura 4.3: Exemplul de informație de la serverul FTP

Pe lângă pagina principală utilizatorul are acces și la alte pagini. Aceste pagini conțin grafice cu informațiile preluate de la agenții SNMP. În Figura 4.3 este prezentat un astfel de grafic cu memoria RAM disponibilă a serverului FTP. În acest grafic, pe lângă memoria RAM disponibilă, este prezentat și intervalul orar în care au fost preluate aceste date.
Aceste grafice sunt prelucrate cu ajutorul fișierului *snmp-pages.py*. El prelucrează datele trimise de agent către manager, ce sunt salvate inițial într-o bază de date RRD și le transformă în grafice cu ajutorul RRDtool. Acest lucru a fost preferat pentru a putea fi mai ușor de urmărit obiectul monitorizat.

4.3. Probleme întâmpinate și modalități de rezolvarea

După ce a fost configurată aplicația, prima problemă întâmpinată a fost legată de comunicare între obiectul monitorizat și serverul SNMP. Aceasta a fost rezolvată în doi pași. Primul pas a fost realizarea unui fișier xml, care va fi folosit ca și excepție în firewall-ul sistemului de operare de pe serverul SNMP. Codul sursă acestui fișier se găsește în Anexa 2.

După realizarea acestui pas, managerul nu primea informații de la agenți. Acest lucru a fost remediat prin dezactivarea SELinux.

4.3.1. SELinux

SELinux [19], denumit și *Security Enhanced Linux*, reprezintă un mod de îmbunătățire a securității distribuțiilor bazate pe Linux, fiind implementat ca un modul de securitate (Linux Security Module). A fost creat de Agenția Națională pentru Securitate, SELinux se bazează pe tehnica Mandatory Access Control, care permite implementarea unei mari varietăți de polițe de securitate. Pentru a putea folosi această tehnică, modulul SELinux adaugă o referință către kernel-ul (nucleul) Linux-ului, cu ajutorul căreia implementează MAC-ul.

Principalul pe care se bazează această tehnică este acela de a refuza execuția aplicațiilor ale căror permisiuni nu sunt explicit descrise, prin urmare nu se știe exact, care va fi efectul executării lor. Un alt avantaj al SELinux este acela că oferă posibilitatea de a seta anumite limite ale comportamentului unor aplicații ce se încadrează în categoria color potențial dăunătoare sistemului de Linux. Astfel, respectivele aplicații nu pot ieși din acel model comportamental, rămânând blocate și probabil oprite.

Alte probleme întâmpinate au fost legate de serverul web, dar și de necesitatea de a rula două fișiere *snmp-manager.py* și *snmp-pages.py*, la un interval regulat.

Problema legată de serverul web a fost că nu mai mergea serviciul ce are legătură cu pagina web. După o citire amănuntă a informațiilor găsite în log-uri, a trebuit să specific în fișierul de configurare a serverului web ServerName-ul (numele sau eticheta serverului web prin care poate fi accesat pe un browser). Deoarece nu am creat un domeniu, am scris adresa IP a serverului web și problema a fost rezolvată.

Necesitatea de a rula două fișiere la un interval regulat (în acest caz am ales un minut) a fost împlinită prin folosirea Crontab. După realizarea acestui lucru, a apărut altă problemă, în urma executării fișierului *snmp-pages.py* rezulta o eroare cum că nu există fișierul *template.html*. Acest lucru a fost rezolvat, prin forțarea crontab-ului de a deschide mai întâi calea la care există fișierele necesare rulării proiectului și apoi executând acele fișiere. Datele care se găsesc în crontab, se găsesc la Anexa 3.

Capitolul 5. Testarea aplicației și rezultatele experimentale

5.1. Elemente de configurare

Pentru ca proiectul să ofere datele dorite este nevoie ca atât managerul, cât și agenții să fie configurați corespunzător.

5.1.1. Configurarea managerului SNMP

Înainte de a ajunge la configurarea managerului SNMP a fost nevoie de instalarea NetSNMP. După aceea a fost nevoie de configurarea fișierului *snmpd.conf*, care se găsește la calea /var/www/html.
În acest fișier a trebuit să am în vedere versiunea SNMP pe care urma să o utilizez, community string-ul (o mică metodă de securitate, pentru nu a avea oricine acces la datele trimise de agent către manager) și la ce gamă de informații poate avea acces.

În final, am creat scripturile prezentate anterior, pentru o comunicare cât mai ușoară între agent și manager.

Pentru o siguranță se verifică dacă managerul comunică cu agenții prin portul 161. Aceasta se face prin comanda *netstat -tulpan grep 161*.

5.1.2. Configurarea agentului SNMP pe Linux

Asemenea ca pe mașina pe care se găsește managerul SNMP, trebuie realizat și pe obiectele ce se doresc monitorizate (mai puțin partea cu realizarea scripturilor).

Informațiile ce se regăsesc în fișierul *snmpd.conf* se găsesc la Anexa 4. Ca și precizare, a ultimei linii ,aflată în fișierul *snmpd.conf*, este necesară pentru a trimite date despre spațiu pe disk către manager.

Pe lângă adaptarea fișierului *snmpd.conf* la cele necesare, mai trebuie adăugat excepție pe portul *161/udp* în firewall.

În final, se realizează aceeași verificare, conform căreia agentul comunică cu managerul pe portul 161.

5.1.3. Configurarea agentului SNMP pe Windows

Pe sistemele de operare puse la dispoziție de Microsoft lucrurile sunt puțin mai simple. Aici voi explica pas cu pas cu ajutorul capturilor realizate în timpul configurării.

Fie că este un sistem de operare dedicat uzului de zi cu zi (Windows 7, 8 sau 10) sau dedicat pentru servere, pașii de configurare sunt la fel, cu o singură excepție insignifiantă.

Pasul inițial este să deschizi *Control Panel*, apoi în *Programs and Features* și deschizi *Turn Windows features on or off*. Diferența de care spuneam se găsește aici. Pe sistem de operare dedicat uzului de zi cu zi se deschide o fereastră nouă, ca în Figura 5.1.
În cazul unui sistem de operare dedicat serverului, se va deschide *Server Manager* și *Add Roles and Features Wizard*, ca în Figura 5.2. Aici, se va selecta de la *Features*, *SNMP Service*.

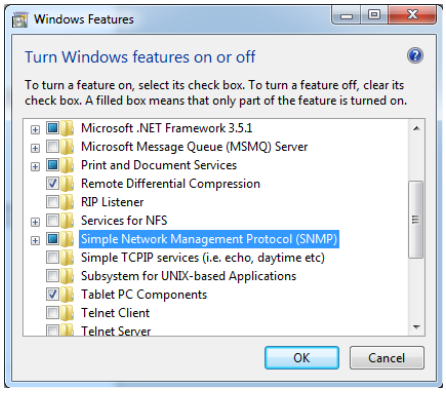

Figura 5.1: Instalare agent SNMP pe Windows 7

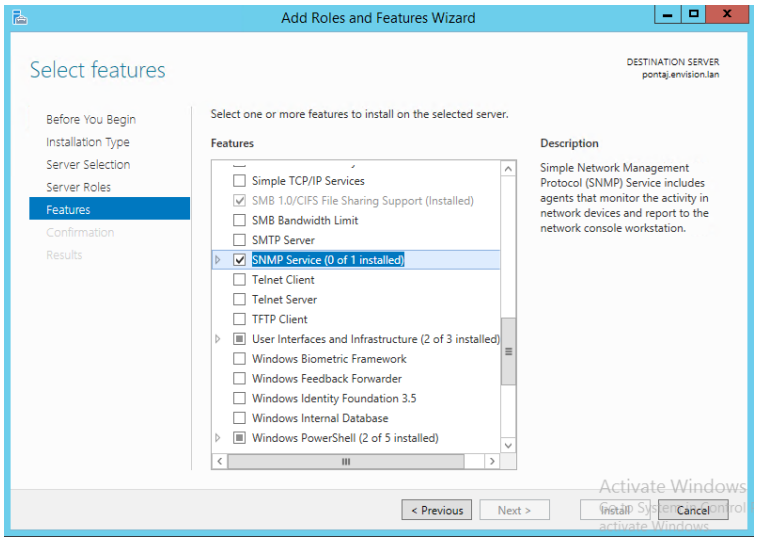

Figura 5.2: Instalare agent SNMP pe Windows Server 2012 R2

După instalarea serviciului SNMP, se va intra la servicii pentru a îl configura. Principalul lucru ce trebuie realizat, este să specifici în tabul de securitate community string-ul și de la cine accepți pachete SNMP. Un exemplu concret al acestor specificații se găsește în Figura 5.3. De asemenea trebuie specificate ce informații să comunice agentul managerului. Acestea se găsesc în tabul Agent. Un exemplu concret se găsește în Figura 5.4.

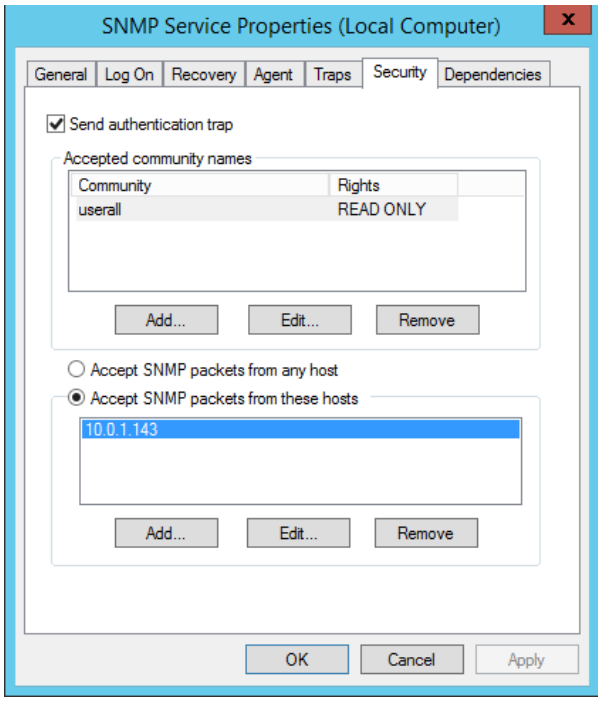

Figura 5.3: Exemplu de configurare în tabul Securitate

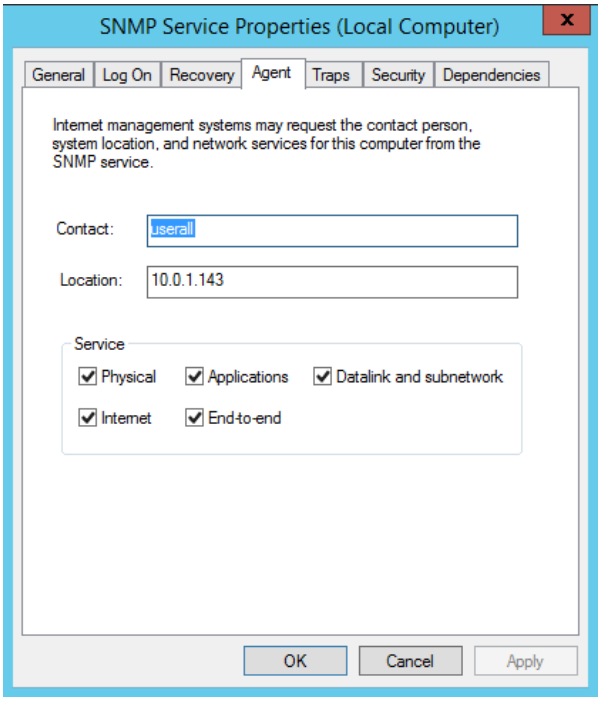

Figura 5.4: Exemplu de configurare în tabul Agent

La finalul acestor configurări, trebuie restartat serviciul SNMP pentru a lua în considerare modificările efectuate.

5.2. Testarea sistemului

Testarea sistemului a fost realizată pe serverul SNMP. S-a adaptat fișierul de configurare *snmp-manager.cfg* astfel încât să se monitorizeze serverul SNMP.
Pentru a genera câteva rezultate a fost nevoie să utilizăm resursele mașinii, prin urmare am deschis un browser și am navigat pe internet, timp de 10 minute. În final, s-au observat graficele generate pentru traficul de download, upload, memoria RAM disponibilă și procentul CPU utilizat. De asemenea pentru memoria RAM și CPU am făcut o comparație cu datele obținute din aplicația top și acestea coincideau cu graficul generat pe pagina web.

Pentru testarea comunicării dintre agent și manager, am folosit comanda *snmpwalk*. Cu ajutorul acestei comenzi făceam cerere la agent să îmi genereze date, în funcție de OID-ul specificat. Aceste date erau furnizate la consolă.

Un exemplu concret al acestei comenzi este următorul:

smnpwalk -v 2c -c userall 10.0.1.131 1.3.6.1.2.1.2.2.1.10.

Primul lucru specificat după *snmpwalk* este versiunea (-v), care în exemplul de față este 2c. Al doilea lucru ce era necesar pentru a primi date de la agent era community string-ul (-c), care este *userall*. Ultimele două lucruri sunt adresa IP a obiectului de la care se doresc informații (10.0.1.131) și OID-ul *1.3.6.1.2.1.2.2.1.10*. Cu ajutorul acestui OID am primit informații de la toate plăcile de rețea existente, în legătură cu traficul pentru download. Pentru a face distincția între plăcile de rețea a trebuit să folosesc alt OID și anume *1.3.6.1.2.1.2.2.1.2*. Cu ajutorul acestui OID îmi afișează pe consolă o descriere succintă a fiecărei plăci de rețea existente (în acest caz era 23 plăci de rețea).

Concluzii

După cum mi-am propus, înainte de a proiecta aplicația, am reușit să îndeplinesc ceea ce mi-am propus și anume să observ, în timp real, starea de funcționare a echipamentelor de comunicație sau a echipamentelor destinate anumitor servicii.

Acest proiect a fost o provocare pentru mine. Pe de o parte, am învățat destul de multe lucruri și mi-a trezit interesul pentru a continua în acest domeniul. Pe de altă parte, am întâmpinat destul de multe dificultăți, dar cu toate acestea sunt bucuros de ceea ce am realizat.

Proiectul poate fi dezvoltat în continuare. El ar putea să conțină mai multe date statistice de la obiectele monitorizate, folosirea *wake on LAN* pentru a putea porni un calculator, posibilitatea de a restarta diferite servicii sau placa de rețea și adăugarea unui nou obiect ce se dorește monitorizat direct din pagina web. Pe lângă acestea, proiectul ar mai putea include o metodă de autentificare (de preferat prin LDAP), un nume de domeniu și un certificat pentru HTTPS.

Ca și concluzie generală, acest proiect poate fi de ajutor oricărui administrator de sistem, care dorește să folosească o soluție proprie.

Bibliografie

[1] Douglas Mauro, Kevin Schmidt, „Essential SNMP”, O'Reilly, 2008.

[2] Aaron Leskiw, SNMP Tutorial Part 2: Rounding Out the Basics [Online], Disponibil la adresa: http://www.networkmanagementsoftware.com/snmp-tutorial-part-2-rounding-out-the-basics/, Accesat: 2016.

[3] Kevin R. Fall, W. Richard Stevens, „The Protocols”, Addison-Wesley Professional Computing Series, 2012.

[4] William Stallings, „SNMP, SNMPv2, SNMPv3 and RMON 1 and 2”, Addison-Wesley Professional, 1993.

[5] , RRDtool [Online], Disponibil la adresa: http://oss.oetiker.ch/rrdtool/, Accesat: 2016.

[6] , De ce Python [Online], Disponibil la adresa: http://py4school.rosedu.org/wiki/doku.php?id=intro-si-sintaxa, Accesat: 2016.

[7] , Why Django [Online], Disponibil la adresa: https://www.djangoproject.com/start/overview/, Accesat: 2016.

[8] , [Online], Disponibil la adresa: http://jinja.pocoo.org/, Accesat: 2016.

[9] , About Apache [Online], Disponibil la adresa: http://httpd.apache.org/ABOUT_APACHE.html, Accesat: 2016.

[10] , [Online], Disponibil la adresa: https://www.nagios.com/, Accesat: 2016.

[11] , What is Cacti? [Online], Disponibil la adresa: http://www.cacti.net/what_is_cacti.php, Accesat: 2016.

[12] , About Zenoss [Online], Disponibil la adresa: https://www.zenoss.com/company, Accesat: 2016.

[13] Andrew Tanenbaum, „Rețele de calculatoare”, Byblos, 2003.

[14] Rand Morimoto, Michael Noel, Guy Yardeni, Omar Droubi, Andrew Abbate, Chris Amaris ; technical edit by Tyson Kopczynski, „Windows Server 2012 unleashed”, SAMS, 2013.

[15] Aidan Finn, Patrick Lownds, Michel Luescher, Patrick Lownds, Damian Flynn, „Windows Server 2012 Hyper-V Instalation and Configuration Guide”, SYBEX, 2013.

[16] , Ce este Linux? [Online], Disponibil la adresa: http://www.linux.ro/, Accesat: 2016.

[17] , What is CentOS Linux? [Online], Disponibil la adresa: https://wiki.centos.org/?id=3, Accesat: 2016.

[18] Dina Glazer, SNMP Tutorial Part 4: Layered Communication [Online], Disponibil la adresa: http://userpages.umbc.edu/~dgorin1/451/snmp/snmp_tutorial.htm, Accesat: 2007.

[19] Christopher Negus, „Linux Bible”, Wiley, 2015.

Anexe

Anexa 1. Exemplul de informații din fișierul snmp-manager.cfg

```
[system_4]
description=Server FTP
address=10.0.1.65
port=161
communityro=userall

[check_40]
description=WLAN incoming traffic
oid=1.3.6.1.2.1.2.2.1.10.2
system=system_4
sampling_rate=100

[check_41]
description=WLAN outgoing traffic
oid=1.3.6.1.2.1.2.2.1.16.2
system=system_4
sampling_rate=100

[check_42]
description=Available Disk Space
oid=1.3.6.1.4.1.2021.9.1.7.1
system=system_4
sampling_rate=600

[check_43]
description=Percentage of User CPU time
oid=1.3.6.1.4.1.2021.11.9.0
system=system_4
sampling_rate=600

[check_44]
description=RAM Available
oid=1.3.6.1.4.1.2021.4.6.0
system=system_4
sampling_rate=600
```

Anexa 2. Script xml pentru excepție firewall

```
<?xml version=”1.0” encoding=”utf-8”?>
<service>
	<short>SNMP</short>
	<description>SNMP protocol</description>
	<port protocol=”udp” port=”161”/>
</service>
```

Anexa 3. Executarea unui fișier la un interval regulat de timp

```
*/1 * * * * (cd /var/www/html/ ; ./snmp-manager.py >> log.txt)
*/1 * * * * (cd /var/www/html/; /snmp-pages.py >>log.txt)
```

Anexa 4. Configurarea fișierului snmpd.conf

```
# Map 'userc'   community to the 'ConfigUser'
# Map 'userall' community to the 'AllUser'

#            sec.name       source         community
com2sec ConfigUser     default        userc
com2sec AllUser        default        userall
# Map 'ConfigUser' to 'ConfigGroup' for SNMP Version 2c
# Map 'AllUser' to 'AllGroup' for SNMP Version 2c

#                        sec.model      sec.name
group   ConfigGroup    v2c            ConfigUser
group   AllGroup       v2c            AllUser

# Define 'SystemView', which includes everything under .1.3.6.1.2.1.1 (or .1.3.6.1.2.1.25.1)
# Define 'AllView', which includes everything under .1
#                        incl/excl      subtree
view    SystemView     included       .1.3.6.1.2.1.1
view    SystemView     included       .1.3.6.1.2.1.25.1.1
view    SystemView     included       .1.3.6.1.2.1.2.2.1.10
view    SystemView     included       .1.3.6.1.2.1.2.2.1.16
view    SystemView     included       .1.3.6.1.4.1.2021.9
view    AllView        included       .1.3.6.1.4.1.2021.9
view    AllView        included       .1

# Give 'ConfigGroup' read access to objects in the view 'SystemView'
# Give 'AllGroup' read access to objects in the view 'AllView'
#                        context model  level        prefix    read           write   notify
access  ConfigGroup    ""      any    noauth       exact     SystemView     none    none
access  AllGroup       ""      any    noauth       exact     AllView        none    none

disk / 100000
```