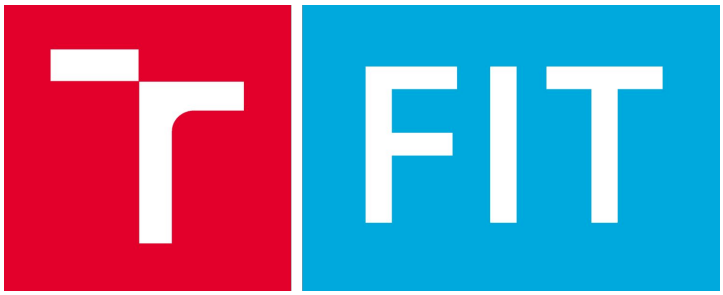
VYSOKÉ UČENIE TECHNICKÉ V BRNE

Fakulta informačných technológií



Sieťové aplikácie a správa sietí

2015/2016

Programovanie sieťovej služby

Jednoduchý SNMP Manager

Filip Gulán (xgulan00) 20.10.2015

Obsah

1 Úvod

Úlohou projektu, ktorý popisuje táto dokumentácia, bolo vytvoriť jednoduchý SNMP Manager pre vyčítanie parametrov rozhraní zo SNMP agenta. Manager má z určeného agenta vyčítať parametre objektu s OID 1.3.6.1.2.1.2. Program má tieto parametre vyčítať periodicky v určenom intervale a vypisovať ich na štandardný výstup. Ukončiť sa má korektne po obdržaní signálu SIGINT.

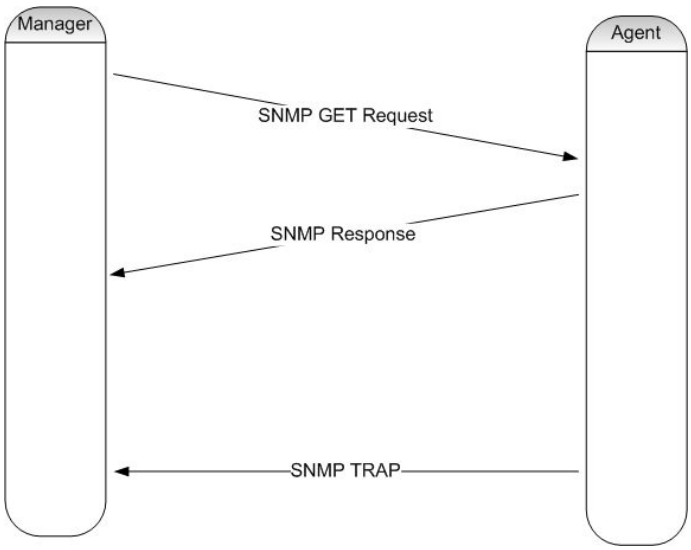
Testovanie je doporučené vykonávať na poskitnutom virtuálnom stroji s operačným systémom Ubuntu 14.04 LTS, ktorý je vytvorený špeciálne pre testovacie účely projektu.

2 Úvod do problematiky

Skôr ako začnem popisovať svoje vlastné riešenie, vysvetlím niekoľko termínov, ktoré je nutné poznať pre pochopenie danej problematiky projektu.

2.1 SNMP

SNMP, alebo Simple Network Management Protocol je rozšírený a štandardizovaný internetový protokol slúžiaci na vyčítanie hodnôt zo zariadení, napríklad smerovačov, prepínačov, pracovných staníc, tlačiarní a iných. SNMP pracuje v aplikačnej vrstve, teda 7. vrstve OSI modelu. Agent prijíjma požiadavky na UDP porte 161 (nezabezpečený SNMP) a porte 10161 (v prípade zabezpečeného SNMP). Manager prijíma asynchrónne požiadavky od agenta na porte 162 (nezabezpečené SNMP), alebo 10162 (zabezpečené SNMP). Manager posiela požiadavky na agenta z akéhokoľvek dostupného portu a agent odosiela odpoveď späť managerovi na ten istý port, z ktorého požiadavka prišla. Sieť riadená SNMP sa typicky skladá z 3 častí: spravované zariadenia, software, ktorý beží na spravovaných zariadeniach (agent) a nakoniec software, ktorý beží na správcovských zariadeniach.

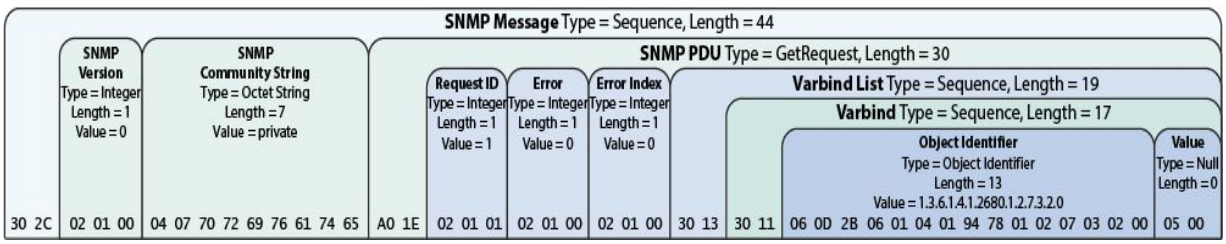


Obrázok 1. - Príklad jednoduchej komunikácie SNMP Agenta a Managera
zdroj: http://www.racom.eu/eng/products/m/ripex/app/snmp.html

2.2 SNMP datagram

SNMP formát správy udáva, aké položky majú byť zahrnuté v správe, a v akom poradí. Celá správa sa skladá z niekoľkých vnorených vrstiev. Každá vrstva, alebo položka obsahuje typ, dĺžku a hodnotu. Zoberme si napríklad vrstvu SNMP Message z obrázku číslo 2. Môžeme vidieť že má typ Sequence, dĺžku 44 a jej value, alebo dáta sa skladajú z ďalších vrstiev, v tomto prípade SNMP version, SNMP Community String a SNMP PDU. Aby sa jednalo o SNMP datagram a agent vedel na našu požiadavku odpovedať, tak v datagrame musia byť prítomné všetky vrstvy/položky.
Popis položiek SNMP správy:

- **SNMP Message -** sekvencia reprezentujúca celú správu SNMP.
- **SNMP Version -** číslo reprezentujúce verziu SNMP.
- **SNMP Community String -** reťazec použitý pri zabezpečení SNMP agenta.
- **SNMP PDU -** obsahuje telo SNMP správy.
- **Request ID -** integer identifikujúci konkrétny SNMP požiadavok.
- **Error -** integer udávajúci chybu, ak nenastala chyba tak obsahuje 0x00, v prípade chyby 0x001 - 0x005.
- **Error Index -** v prípade nastatia chyby drží ukazateľ na objekt, ktorý chybu spôsobil.
- **Varbind List -** sekvencia varbindov.
- **Varbind -** sekvencia 2 polí, object identifier a value.
- **Object Identifier -** id konkrétneho objektu v agentovi.
- **Value -** hodnota, líši sa podľa SNMP PDU typu v prípade GetRequest napríklad vracia dáta získané zo SNMP agenta.



Obrázok 2: Popis štruktúry SNMP paketu
zdroj: http://www.rane.com/note161.html

3 Popis vlastného riešenia

Nasledujúce riadky popisujú moje vlastné riešenie problematiky, návrh riešenia, popis implementácie, popis programu a návod na použitie.

3.1 Návrh riešenia

SNMP požiadavky na agenta som sa rozhodol posielať po jednej a typ dátovej jednotky som zvolil getRequest. Skúšal som aj možnosť poslať jednu požiadavku obsahujúcu všetky oid, ale tu nastal problém. Zatiaľ čo v laboratórnych podmienakch všetko fungovalo, tak na reálnom zariadení, kde chýbal napríklad objekt 9, my v danej odpovedi v poli value pre všetky, aj tie objekty, ktoré existovali na danom zariadení, neprišli žiadne dáta. Skúšal som to s rôznymi kombináciami, ale ako

som zistil, vždy ak som poslal hromadnú požiadavku na agenta a vyskitoval sa tam neexistujúci objekt, tak som dostal dáta všetky typu null, teda žiadne. Z toho dôvodu som sa rozhodol požiadavky posielať oddelene, aj vzhľadom k celkovej menšej rýchlosti programu.

3.2 Spracovanie argumentov

Argumenty sa spracovávajú v hlavnej funkcií *main()*. Rozhodol som sa použiť klasickú metódu spracovania argumentov, pomocou *int argc* a *char* argv[]*, kvôli malé počtu argumentov a hlavne kvôli, z môjho pohľadu, atypicky umiestnenému argumentu AGENT.

3.3 Komunikácia zo SNMP agentom

Komunikácia so SNMP agentom prebieha vo funkcií *komunikacia()*, kde najprv vytvorím a pripravím UDP datagram a potom v nekonečnom cykle odosielam vytvorený datagram spolu s mnou vytvorenou SNMP vrstvou (SNMP vrstva sa vytvára v 2 vnorených for cykloch, viz. ďalej). Odosielam prostredníctvom knižnej funkcie *sendto()* a následne prijímam odpoveď prostredníctvom funkcie *recvfrom()*. Na konci cyklu sa nachádza funkcia *usleep()*, ktorá zabezpečuje odosielanie požiadavkov periodicky, podľa zadaného intervalu, ktorý je špecifikovaný argumentom -i/--interval.

Vo vnútri nekonečného cyklu sa nachádzajú 2 ďalšie vnorené for cykle. Prvý for prechádza rozhrania, a druhý for prechádza parametre objektu 1-22. Prvý priechod prvého for cyklu slúži na zistenie počtu rozhraní, kedy sa namiesto parametrov objektu s OID 1.3.6.1.2.1.2 dotazujem na objekt s OID 1.3.6.1.2.1.2.1, v ktorom je uložený počet rozhraní. Pri ďalšom priechode podľa týchto cyklov generujem OID, na ktoré sa chcem dotazovať a následne sa vytvára celá SNMP správa vo funkcií *urobGetRequest()*.

3.4 Vytváranie SNMP správy na odoslanie

SNMP správa sa vytvára vo funkcií *urobGetRequest()*, ktorá ako argument prijíma community string, a oid, ktorého hodnotu, chceme získať od SNMP agenta. Vo funkcií sa postupne vytvára hexa string, podľa datagramu na obrázku 3. Postupujeme odzadu, a teda najprv vytvoríme položku value, ktorá je vždy prázdna, teda dĺžku bude mať 0x00 a typ NULL, teda 0x05 a pridáme tento string “0500” na koniec prázdneho pomocného stringu. Potom vytvoríme položku object identifier, kde najprv na začiatok pomocného stringu pridáme oid, získaného ako druhý argument funkcie, potom spočítame jeho dĺžku a následne túto dĺžku v ako hexa string (použijeme našej funkcie intNaHex) pridáme na začiatok nášho pomocného stringu. Nakoniec pridáme znova na začiatok pomocného strignu typ oid teda 0x06. A takto sa postupuje smerom doprava, aby sme vytvorili kompletnú SNMP správu.

3.5 Prečítanie príchodzej SNMP správy a vyčítanie hodnoty

Príchodzá správa sa dekóduje vo funkcií *dekodujGetResponse()*. Funkcia postupne rozpoznáva dané polia a postupne ich vymazáva, až dokým nenarazí na pole Value. Na obrázku 3. postupujeme zprava doľava. Funkcia *dekodujGetResponse() potom value* vráti a typ a jednotlivé dáta value sa získavajú vo funkcií *ziskajValue()*. Funkcia sa na základe typu rozhoduje, ako majú byť dané dáta reprezentované. Táto funkcia potom už value hodnotu vráti vo formáte stringu v požadovanej reprezentácií.

3.6 Rozšírenie

Súčasťou projektu je aj vypracované rozšírenie. Keďže viem, čo presne mi má prísť, tak viem aj skontrolovať, či mi prišlo presne to čo som požadoval. Rozšírenie je implementované vo funkcií *dekodujGetResponse(),* kde kontrolujem formát celej príchodzej správy. Kontrolujem všetky typy či súhlasia a aj celé oid a commnity string. V prípade zistenia, že je formát nesprávny, ukončím program s chybovou hláškou.

3.7 Popis a návod na použitie programu

Program sa spúšťa s 2 povinnými parametrami a jedným nepovinným. Každý parameter, môže byť zadaný v krátkej, alebo dlhej forme.
Parametre: [-i interval] -c community_string AGENT
-i, --interval: interval medzi jednotlivými vyčítaniami hodnôt v ms
-c, --community_string: community string pre prístup k parametrom na SNMP agentovi
AGENT: doménové meno alebo IP adresa SNMP agenta
Príklad spustenia: ./isaSnmpIfLog -i 500 -c public 127.0.0.1

4 Záver

Program bol riadne otestovaný na operačnom systéme Elementary OS 0.3 Freya a na Ubuntu 14.04 LTS, a na SNMP agentoch umiestnených na localhoste Ubuntu, VUT siete a na osobnom domácom WIFI smerovači značky Asus RT-N16. Behom testovania sa nevyskitli žiadne chyby a všetky navrhnuté testy dopadli úspešne.

Program bol tvorený v programovacom jazyku C++ a vývojovom prostredí CLion od firmy JetBrains, a prekladaný prekladačom g++ pomocou mnou vytvoreného Makefile súboru, ktorý je súčasťou odovzdaného archívu. Behom implementácie programu som pre vizualizáciu správ používal program Wireshark.

Referencie

[1] BRUEY, Douglas. SNMP: Simple? Network Management Protocol [online]. 2005 [cit. 2015-10-20]. Dostupné z: http://www.rane.com/note161.html

[2] KLAŠKA, Luboš. 2000. SNMP objekty a MIB [online]. [cit. 2015-10-20]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=SNMP-objekty-a-MIB-1362000

[3] ZOBEL, Daniel. 2010. SNMP, MIBs and OIDs—an Overview [online]. [cit. 2015-10-20]. Dostupné z: http://kb.paessler.com/en/topic/653-how-do-snmp-mibs-and-oids-work

[4] Simple Network Management Protocol. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-10-20]. Dostupné z: https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol