**UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA**
**PRÍRODOVEDECKÁ FAKULTA**

# BEZPEČNOSTNÝ MONITORING SIEŤOVÝCH PREPÍNAČOV V LOKÁLNEJ POČÍTAČOVEJ SIETI

**2011** **Ján Host**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA
PRÍRODOVEDECKÁ FAKULTA

# BEZPEČNOSTNÝ MONITORING SIEŤOVÝCH PREPÍNAČOV V LOKÁLNEJ POČÍTAČOVEJ SIETI

BAKALÁRSKA PRÁCA

| | |
|---|---|
| Študijný program: | Informatika |
| Pracovisko (katedra/ústav): | Informatika |
| Vedúci diplomovej práce: | Mgr. Pavol Sokol |

Košice 2011 **Ján Host**

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Ján Host
**Študijný program:** Informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
**Študijný odbor:** 9.2.1. informatika
**Typ záverečnej práce:** Bakalárska práca
**Jazyk záverečnej práce:** slovenský

**Názov:** Bezpečnostný monitoring sieťových prepínačov v lokálnej počítačovej sieti

**Cieľ:** (1) Zozbierať a prezentovať známe prostriedky týkajúce sa bezpečnostného monitoringu sieťových prepínačov použitím protokolu SNMP. Analyzovať výhody a riziká používania.

(2) Navrhnúť a implementovať bezpečnostný monitoring siete sieťových prepínačov v prostredí reálnej sieťovej prevádzky.

**Literatúra:**
[1] FRY,C., NYSTROM,M.: Security Monitoring. 1 vyd. O'Reilly, 2009. 256 s. ISBN-13: 9780596518165
[2] STALLINGS,W.: Network Security Essentials. Application and Standards. 4. vyd. Prentice Hall, 2010. 432 s. ISBN-13: 9780136108054
[3] Barth,W.: Nagios. Systems and Network Monitoring. 2 vyd. San Francisco: No Starch Press, 2008. 720 s. ISBN-13: 9781593271794
[4] SCHRODER,C.: Linux Networking Cookbook. O'Reilly, 2007. 640 s. ISBN-13: 9780596102487

**Anotácia:** Základným sieťovým zariadením v lokálnych počítačových sieťach sú sieťové prepínače. Práca sa zaoberá bezpečnostným monitoringom týchto zariadení.

**Kľúčové slová:** monitoring počítačovej siete, sieťový prepínač, SNMP, bezpečnosť

**Vedúci:** Mgr. Pavol Sokol
**Ústav:** ÚINF - Ústav informatiky

**Dátum zadania:** 30.09.2011

**Dátum schválenia:** 30.09.2011

prof. RNDr. Viliam Geffert, DrSc.
riaditeľ ústavu

Univerzita Pavla Jozefa Šafárika v Košiciach
Prírodovedecká fakulta
Ústav informatiky

## Vyhlásenie

Vyhlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

.....

*Podpis*

Poďakovanie

Rád by som sa poďakoval vedúcemu svojej práce, Mgr. Pavlovi Sokolovi za prejavenú podporu a pomoc pri vedení tejto práce a svojej rodine, ktorá mi verila.

**Abstrakt**

Bakalárska práca sa zaoberá teoretickým opísaním možností monitorovania počítačových sietí v reálnej prevádzke, a hlavne návrhom a implementáciou vlastného monitorovacieho nástroja. Teoretická časť práce je venovaná rôznym dostupným monitorovacím nástrojom, ktoré sú bežne používané správcami sietí, ale aj bežnými užívateľmi. Práca definuje a približuje SNMP protokol a jeho spektrum použitia, android layout a android virtual device a SNMP4J knižnicu, pomocou ktorých sa následne vyvíjala aplikácia. V praktickej časti je v jazyku JAVA navrhnutý a implementovaný základný monitorovací nástroj pre operačný systém android pre sieťové prepínače s podporou SNMP protokolu. Výsledkom práce je aplikácia s názvom Switch Info, ktorá spĺňa vyššie uvedené požiadavky.

**Kľúčové slová:** počítačové siete, monitorovacie nástroje, SNMP protokol, android layout, SNMP4J knižnica

**Abstract**

The present bachelor's thesis concentrates on the theoretical description and practical solutions of monitoring computer networks in their real operation. The theoretical part of the work looks at various monitoring tools that are available to both – network administrators and users. Based on the theoretical concepts it draws upon (monitoring tools, SNMP protocol, android layout, SNMP4J library), it designs and implements a new monitoring tool. In the practical part of the thesis, the author, using Java programming language, designed and implemented a basic monitoring tool for the operating system android for network switches that support the SNMP protocol. The final product of the present bachelor's thesis is a tool named Switch Info.

**Keywords:** computer networks, monitoring tools, SNMP protocol, android layout, SNMP4J library

# Obsah

# Zoznam skratiek a značiek

| | |
|---|---|
| NMS | **N**etwork **M**anagement **S**ystem |
| SNMP | **S**imple **N**etwork **M**anagement **P**rotocol |
| GPL | GNU **G**eneral **P**ublic **L**icense |
| SMTP | **S**imple **M**ail **T**ransfer **P**rotocol |
| POP3 | **P**ost **O**ffice **P**rotocol |
| HTTP | **H**yper**T**ext **T**ransfer **P**rotocol |
| NNTP | **N**etwork **N**ews **T**ransfer **P**rotocol |
| ICMP | **I**nternet **C**ontrol **M**essage **P**rotocol |
| CPU | **C**entral **P**rocessor **U**nit |
| SSH | **S**ecure **Sh**ell |
| SSL | **S**ecure **S**ockets **L**ayer |
| RRD | **R**ound-**R**obin **D**atabase |
| PHP | **H**ypertext **P**re**p**rocessor |
| IPMI | **I**ntelligent **P**latform **M**anagement **I**nterface) |
| XMPP | E**x**tensible **M**essaging and **P**resence **P**rotocol |
| DHCP | **D**ynamic **H**ost **C**onfiguration **P**rotocol |
| OID | **O**bject **Id**entifier |
| MIB | **M**anagement **I**nformation **B**ase |
| CMIP | **C**ommon **M**anagement **I**nformation **P**rotocol |
| CMIS | **C**ommon **M**anagement **I**nformation **S**ervice |
| GDMO | **G**uidelines for the **D**efinition of **M**anaged **O**bjects |
| UDP | **U**ser **D**atagram **P**rotocol |
| RFC | **R**equest **f**or **C**omments |
| DNS | **D**omain **N**ame **S**ystem |
| TTL | **T**ime **T**o **L**ive |
| CDP | **C**isco **di**scovery **p**rotocol |
| DoS | **D**enial **o**f **S**ervice |

# Úvod

V dnešnej modernej a pretechnizovanej dobe je pre mnohých ľudí nepredstaviteľné, aby vykonávali svoju prácu, študovali a často trávili aj voľný čas bez použitia počítača a internetu, ktorý predstavuje verejne dostupný celosvetový systém vzájomne prepojených počítačových sietí. Málokto si však uvedomuje, aké obtiažne a neustále náročnejšie je udržiavať akúkoľvek počítačovú sieť bezpečnú a plynulú. Táto bakalárska práca sa zaoberá práve bezpečnosťou takejto siete, presnejšie odvetvím bezpečnosti, a to monitoringom. Monitorovanie siete je najdôležitejšia súčasť bezpečnosti, pretože s jeho pomocou dokážeme predvídať problémy a predchádzať im. Monitorovacích nástrojov je neprehľadné množstvo, preto cieľom práce je opísať a priblížiť tie najdôležitejšie a najbežnejšie z nich. Týmito nástrojmi sa zaoberá prvá kapitola tejto práce. V druhej kapitole sa venujeme protokolom používaným v sieťovej komunikácii. Ide najmä o protokol SNMP, ktorý je najčastejšie využívaný. Ďalšia časť práce sa zameriava na návrhy a implementáciu programu na bezpečnostný monitoring sieťových prepínačov.

Významnú súčasť správy lokálnych počítačových sietí tvorí ich monitorovanie. Táto činnosť predstavuje pre správcu takejto siete každodennú činnosť, bez ktorej by sa pri predchádzaní, resp. už pri riešení problémov nezaobišiel. V práci sa zameriavame na bezpečnostný monitoring sieťových prepínačov, konkrétne manažovateľných sieťových prepínačov. Definujeme, čo predstavuje bezpečnostný monitoring sieťových prvkov a jeho podskupinu v podobe bezpečnostného monitoringu sieťových prepínačov.

# 1. Monitorovanie lokálnej počítačovej siete

Táto kapitola predstavuje úvod problematiky monitorovania lokálnej počítačovej siete. Jej obsahom sa snažíme vysvetliť niekoľko základných pojmov potrebných na pochopenie celej problematiky. V tejto časti sa následne budeme zaoberať protokolmi a prostriedkami pre monitorovanie lokálnej počítačovej siete. V závere kapitoly načrtneme základný rámec pre protokol SNMP.

## 1.1. Úvod do problematiky

**Monitorovanie počítačovej siete** (network monitoring) je dôležitá funkcia siete, ktorá môže ušetriť peniaze pomocou zvýšenia výkonu siete, produktivity zamestnancov a zníženia hardvérových požiadaviek. Monitorovanie počítačovej siete môže byť dosiahnuté použitím rôznych softvérov, alebo použitím kombinácie softvéru a hardvéru. Monitorovať zariadenia je dnes možné pomocou rôznych zariadení, od mobilných telefónov cez sieťové prepínače až po servery.

Na tomto mieste je nevyhnutné zdôrazniť vzťah medzi manažmentom a monitorovaním počítačovej siete. **Manažment počítačovej siete** je širším pojmom ako pojem monitorovanie počítačovej siete a zahŕňa aktivity, metódy, procedúry a nástroje, ktoré sa týkajú fungovania, správy, údržby a zabezpečenia sieťových systémov [1].

Systém na monitorovanie monitoruje vnútornú sieť kvôli chybám , môže ich nájsť a opraviť, prípadne im predchádzať. Netreba si to mýliť so systémami detekcie prieniku do siete. Prakticky každý typ počítačovej siete môže byť monitorovaný. Nezáleží na tom, či je to bezdrôtová sieť, káblová LAN, VPN, WAN. Rozhodnutie, čo vlastne na sieti monitorovať, je veľmi dôležité. Často je kladený dôraz na monitorovanie celej infraštruktúry, čo nie je vždy dobrý nápad. Najlepšie je sa sústrediť na tie zariadenia, ktorých prípadné zlyhanie ovplyvní veľa užívateľov. V našom prípade monitorujeme sieťové prepínače. Na smerovačoch a prepínačoch sa monitoruje dostupnosť, RAM využitie CPU a šírky pásma. Ak je monitorovanie dobre nastavené, je potom jednoduché odhaliť útočníka (pomocou nastavenia portov, alebo ak si všimneme nezvyčajnú aktivitu).

Okrem monitorovania dostupnosti sieťových zariadení a aplikácií, je nutné vykonávať pravidelné bezpečnostné kontroly. Bezpečnostná kontrola môže zahŕňať prevedenie skenovania portov na sieťových zariadeniach hľadajúc neoprávnené použitie alebo otvorené porty, ktoré by nemali byť otvorené. Zatiaľ čo všetky tieto testy by mali byť vykonávané v laboratóriu, je nutné vykonávať pravidelné bezpečnostné kontroly na živej sieti, aby bolo zabezpečené, že všetky systémy pracujú s maximálnou efektivitou, a to pomocou náležitých bezpečnostných opatrení. Niektoré z týchto testov môžu byť vykonané s intrusion detection system (IDS), zatiaľ čo iné testy si vyžadujú špeciálny softvér.[2]

## 1.2. Protokoly pre monitorovanie počítačovej siete

V nasledujúcej kapitole sa zameriavame na protokoly pre manažment počítačovej siete ako jej základnej zložky. Postupne rozoberieme najznámejšie protokoly a hlavnú pozornosť venujeme najznámejším z týchto protokolov.

### 1.2.1. Common Management Information protocol

**Common Management Information protocol** (CMIP) je protokol pre manažment počítačovej siete, ktorý je špecifikovaný pomocou OSI modelu. Je definovaný v in ITU-T Recommendation X.711, ISO/IEC International Standard 9596-1. Poskytuje implementáciu pre servisy definované pomocou CMIS (Common Management Information Service) špecifikovaný v ITU-T Recommendation X.710, ISO/IEC International Standard 9595 a poskytuje komunikáciu medzi aplikáciami network managementu a management agents. CMIS-CMIP je network management protokol definovaný v ISO/OSI network management model.

CMIP modely riadenia informácií, pokiaľ ide o manažované objekty, dovoľujú aj modifikáciu, aj vykonanie akcií na manažovaných objektoch. Manažované objekty sú popísané pomocou GDMO (Guidelines for the Definition of Managed Objects), a môžu byť identifikované s DN (distinguished name). CMIP tiež poskytuje dobrú bezpečnosť (podporuje autorizáciu, kontrolu prístupu, bezpečnostné logovanie) a prispôsobivé reportovanie nezvyčajnej sieťovej aktivity.

### 1.2.2. Simple gateway monitoring protocol

Je to protokol na aplikačnej vrstve, ktorý dovoľuje vzdialeným používateľom kontrolovať a meniť manažovateľné informácie pre bránu. V kontexte internetu, brána vykonáva funkciu klasického sieťového prepínača.

Typické aplikačné prostredie je také, cez ktoré správy prúdia medzi monitorovacími entitami (brány, smerovače, koncové zariadenia servery a iné). Protokol funguje na báze datagramov, ktorých každá správa je kompletne a nezávisle interpretovaná jedinou správou konkrétneho autentifikačného protokolu. Všetky známe implementácie sú postavené na UDP (User Datagram Protocol). [5]

## 1.3. Prostriedky monitorovania počítačovej siete

Táto podkapitola sa zaoberá popisom prostriedkov použitých pri manažmente a monitorovaní počítačovej siete a jej sieťových prvkov. Podstatná časť kapitoly je venovaná najrozšírenejšiemu prostriedku, a to Simple network management protokolu.

### 1.3.1. Windows Management Instrumentation (WMI)

**Windows Management Instrumentation (WMI)** je sada rozšírenie modelu Windows Driver, ktorá poskytuje rozhranie operačného systému, cez ktorý inštruované komponenty poskytujú informácie a notifikácie. WMI je Microsoft implementácia Web-Based Enterprise Management (WBEM) a Common Information Model (CIM) štandardov z Distributed Management Task Force (DMTF). WMI umožňuje skriptovacím jazykom, ako VBScript alebo Windows PowerShell spravovať Microsoft Windows osobné počítače a servery, a to ako lokálne i vzdialene. WMI je predinštalovaný v systéme Windows 2000 a novších. Je k dispozícii na stiahnutie pre Windows NT, Windows 95 a Windows 98. Je teda vhodný len na správu Microsoft zariadení, i keď spolupracuje aj so SNMP.

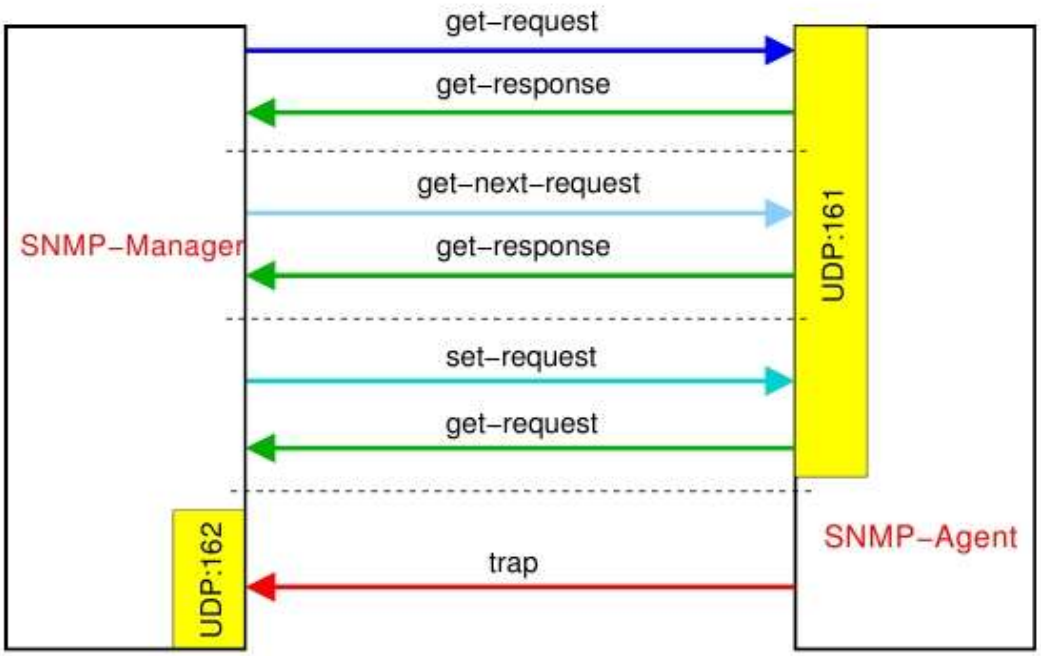
### 1.3.2. Java Management Extensions (JMX)

**Java Management Extensions (JMX)** je Java technológia, ktorá poskytuje nástroje pre správu a monitorovanie aplikácií, systémových objektov, zariadení (napr. tlačiarne) a služieb orientovaných sietí. Tieto zdroje sú reprezentované objektmi volané

MBeans (Managed Bean). V API môže byť dynamicky načítané triedy a inštancie. Správa a sledovanie aplikácií môže byť navrhnutá a vyvinutá v Java Dynamic Management Kit. Je vhodný na monitorovanie Java-based zariadení.[3]

## 1.4. Simple Network Management Protocol

**Simple Network Management Protocol (SNMP)** je protokol, ktorý slúži ako na sledovanie, tak na nastavovanie sieťových zariadení. Používa UDP port 161, vyžaduje pre komunikáciu práve dve strany a pracuje na princípe agenta (server) a správcu (klient). Správca posiela agentovi otázky a príjma odpovede, pri čom jedného agenta sa môže naraz pýtať viac správcov nezávisle na sebe. Agent je vlastne softvér na spravovanom sieťovom zariadení, ktorý zasiela správcovi oznámenia a hodnoty. SNMP umožňuje nie len čítanie hodnôt, ale aj nastavovanie niektorých z nich.




**Obrázok č. 1 – Relácia SNMP protokolu a jej jednotlivé informačné toky**

### 1.4.1. História SNMP

Na začiatku vývoja SNMP bol výbor Internet Architecture Board(IAB1), ktorý ho navrhol ako jednu z možností skôr navrhovaného Simple Gateway Monitoring Protocol(SGMP), ktorý bol vytvorený v roku 1987 a slúžil na komunikáciu medzi

smerovačmi a bránami. Druhý variant bol Common Management Information Protocol over Transmission Control Protocol(CMOT), ktorý bol zameraný objektovo, čo sa mu stalo osudným. Bol zložitejší ako pre užívateľov, tak pre implementáciu na strane výrobcu a záujem oňho výrazne klesol. Ale SNMP sa osvedčil vďaka svojej jednoduchosti a pre výrobcov rôznych zariadení sa postupom času stal štandardnou implementačnou súčasťou. Prvé dokumenty ohľadom SNMP vyšli už v roku 1988 a jednalo sa o špecifikáciu Structure and Identification of Management Information (SMI), Management Information Base (MIB2) a samotného SNMP, ktoré boli v roku 1990 nahradené novšími spracovaniami SMI, MIB a SNMP. Dôležité je, že už tento MIB obsahoval okolo 100 objektov, ktoré sa dali sledovať poprípade nastavovať. V roku 1991 bol vydaný MIB-ll, ktorý definitívne nahradil pôvodný štandard a obsahoval okolo 180 objektov. Pri navrhovaní štruktúry databázy objektov sa počítalo s tým, že sa každú chvíľu bude rozširovať. Dospelo sa k veľmi progresívnemu kroku, a to umožniť výrobcom vytvárať si vlastné sub-špecifikácie, ktoré budú súčasťou úvodnej stromovej štruktúry databázy objektov. Táto myšlienka veľmi spopularizovala SNMP.

### 1.4.2. Vývoj a verzie protokolu

Prvá verzia SNMP (**SNMPv1**) bola schválená v roku 1988 a takmer neriešila bezpečnostné aspekty. Autentizácia správcu voči agentovi bola docielená zaslaním nešifrovaného reťazca community (textový reťazec, ktorý sa tvári ako heslo). Vo väčšine implementácií sa využíva niekoľko reťazcov community ako prístupových hesiel, u každého je daná úroveň oprávnenia. Bezpečnostné vylepšenia a opravy priniesla až druhá verzia **SNMPv2**. Nový spôsob zabezpečenia však pre svoju zložitosť nebol všeobecne prijatý medzi výrobcami. Vzniklo preto niekoľko ďalších štandardizovaných implementácií a postup autentifikácie správcu k agentovi sa tak vo väčšine prípadov nezmenil. Novinkou sa stali iba operácie pre získanie veľkého objemu dát pre odpoveď na jednu požiadavku a vyriešenie bezpečnostných otázok sa odložilo do tretej verzie. **SNMPv3** nakoniec priniesla okrem niekoľkých bezpečnostných modelov aj vylepšenie konfigurácie samotného protokolu na diaľku.

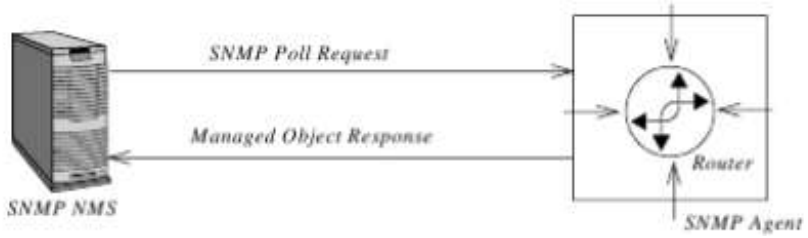
Bezpečnostné politiky a rozmanitá miera implementácie SNMP medzi zariadeniami rôznych výrobcov mali za následok vznik viac súbežne používaných verzií

protokolu. Tie sa líšili prevažne spôsobom autentifikácie a prípadným šifrovaním, ako uvádza nasledujúca tabuľka:

| | |
|---|---|
| SNMPv1 | Prvá verzia, autentifikácia na základe reťazca community |
| SNMPv2p | Druhá verzia protokolu, veľmi zložitá |
| SNMPv2c | Druhá verzia, k autentifikácii používa reťazec community |
| SNMPv2u | Druhá verzia, bezpečnosť založená na užívateľoch |
| SNMPv2* | Nie je štandardizovaná IETF, založená na SNMPv2u |
| SNMPv3 | Tretia verzia, implementuje bezpečnosť založenú na užívateľoch |

### 1.4.3. Management Information Base

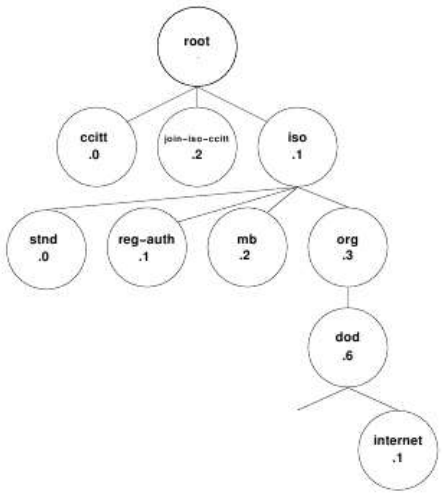
**Management Information Base (MIB)** je databáza manažovateľných objektov, ku ktorým majú prístup manažovacie protokoly. SNMP MIB je množina parametrov, na ktoré sa SNMP manažovacia stanica môže pýtať alebo nastaviť pomocou SNMP agenta sieťového prvku, ako napríklad sieťový smerovač.




**Obrázok č. 2 - Ukážka spojenia SNMP stanice a dopytovaného zariadenia**

Na Obrázku č.2 dáta získané z každého rozhrania, reprezentované šípkou, sú manažovateľné objektové inštancie, ale celkový počet dát zo všetkých štyroch rozhraní je jeden manažovateľný objekt. SNMP obsahuje dve štandardné MIB tabuľky. Prvá, MIB I, definovaná v RFC 1156, bola vytvorená za účelom manažovania na TCP/IP postavenom internetu. MIB II je vlastne len rozšírenie MIB I a je definovaná v RFC 1213. MIB II odkazuje na súčasnú definíciu. SNMPv2 obsahuje MIB II a pridáva nejaké nové objekty. Existujú MIB rozšírenia pre každú množinu prislúchajúcich sieťových prvkov, ktoré môžu byť manažovateľné. Napríklad existujú MIB definície

špecifikované vo forme RFC (Request for Comments) pre AppleTalk, DNS (Domain Name System). Výrobcovia nových zariadení môžu vytvárať a registrovať nové MIB nadstavby (napríklad spoločnosť CISCO).[5, 7]




```
iso.org.dod.internet = .1.3.6.1
```

**Obrázok č. 3 - MIB štruktúra s príkladom**

# 2. Monitorovanie sieťových prvkov z hľadiska bezpečnosti

V tejto kapitole sa zaoberáme používanými monitorovacími systémami, ich výhodami a reálnym použitím. Pojem monitorovanie siete popisuje použitie systému, ktorý neustále monitoruje počítačovú sieť kvôli pomalým alebo kaziacim sa komponentom a upozorňuje správcu siete (pomocou emailu, sms...) na tieto chyby.[2]

## 2.1. Sieťové manažment systémy

**Sieťový manažment systém (NMS)** je kombinácia hardvéru a softvéru používaného na administráciu a monitorovanie počítačovej siete. NMS implementuje viacero protokolov, napríklad SNMP (Simple Network Management Protocol). SNMP je používaný na zber informácií zo sieťových zariadení. Do práce sme vybrali skupinu najpoužívanejších monitorovacích systémov. Hlavnými kritériami ich výberu bola možnosť využitia SNMP protokolu a licenčné podmienky ich použitia. Vyberali sme open-source systémy, ktoré je možné do budúcna modifikovať podľa požiadaviek a stavu počítačovej siete.

### 2.1.1. Nagios

**Nagios** je populárny open source systém na automatické sledovanie stavu počítačových sietí a nimi poskytovaných služieb. Je primárne vyvinutý pre Linux, ale ja možné ho prevádzkovať aj na iných unixových systémoch. Je vydávaný pod GPL (GNU General Public License) licenciou a udržiavaný Ethanom Galstadtom a mnohými ďalšími vývojármi. Pôvodne sa projekt nazýval Netsaint a v roku 2002 sa premenoval na NAGIOS. Je to nástroj, ktorý umožňuje monitorovať počítačovú sieť a v nej poskytované služby a v prípade výskytu problému okamžite informovať administrátora, ktorý tým pádom môže rýchlo zasiahnuť. Monitorovacia služba periodicky spúšťa kontroly špecifikovaných koncových uzlov a služieb. Používa na to externé moduly, ktoré oznamujú výsledok kontroly hlavnému modulu Nagiosu. Ak sa vyskytol problém, služba pošle upozornenie na predvolený kontakt (email, sms, icq...). Aktuálny stav, história záznamov a ďalšie výstupy sú prístupné cez webové rozhranie. Výhody:

monitoruje Windows, Linux, Unix; monitoruje ľubovoľné typy služieb; dobre otestovaný aj vo veľmi zložitých konfiguráciách; pružne prispôsobiteľný a rozšíriteľný; žiadne licenčné poplatky. Nevýhody: vyžaduje server s veľkou pamäťou; konfiguruje sa manuálne, nie je to jednoduché a zaberá to veľké množstvo času. Zvláda monitorovanie sieťových služieb SMTP, POP3, HTTP, NNTP, ICMP, SNMP, monitorovanie systémových prostriedkov (CPU, disk, logovanie- pomocou pluginu aj Windows), vzdialené monitorovanie cez SSH, SSL. Podporuje tiež Event Handling na aktívne riešenie problémov (napríklad automatický reštart služieb).

### 2.1.2. Cacti

**Cacti** je nástroj na tvorbu grafov sieťovej prevádzky, ktorý je dizajnovaný ako frontend k RRDtool (round-robin database tool) backendu. Je vyvíjaný ako intuitívny a jednoducho použiteľný nástroj, ktorý je dobre škálovateľný. Vo väčšine prípadov sa využíva na tvorbu grafov vyťaženia CPU a využitia pásma eternetových rozhraní, ale dá sa použiť aj na monitoring voľného miesta na harddisku alebo počet dopytov za sekundu na MySql server. Frontend je napísaný v jazyku PHP. Zvláda prístup viacerých používateľov, preto je vhodný pre poskytovateľov webových služieb na ukázanie štatistiky šírky pásma zákazníkom. Konfigurácia je jednoduchá.

### 2.1.3. Zabbix

**Zabbix** je NMS vytvorený Alexeiom Vladishevom. Je dizajnovaný na monitorovanie a zaznamenávanie statusu rôznych sieťových služieb, serverov a ďalšieho hardvéru. Používa MySQL, PostgreSQL, SQLite, Oracle alebo IBM DB2 na ukladanie dát. Jeho beckend je naprogramovaný v PHP. Zabbix ponúka viacero monitorovacích možností. Jednoduché kontroly môžu overiť dostupnosť a schopnosť reagovať štandardných služieb ako SMTP alebo http bez inštalácie softvéru na hosťovský počítač. Zabbix agent môže byť nainštalovaný na Unixový alebo Windowsový počítač na monitorovanie štatistiky záťaže CPU, siete, voľného miesta na disku atď. Ako alternatívu voči inštalácii agenta Zabbix ponúka podporu monitorovania cez SNMP, TCP a ICMP, taktiež cez IPMI (Intelligent Platform Management Interface), SSH a telnet. Zabbix podporuje množstvo real-time oznamovacích možností vrátane XMPP (Extensible Messaging and Presence Protocol). Zabbix je tiež vydávaný

pod licenciou GPL. Zabbix začal ako interný softvérový projekt v roku 1998. Po troch rokoch bol vydaný pod GPL licenciou. Vývoj prvej stabilnej verzie trval ďalšie tri roky.

### 2.1.4. MRTG

**MRTG (Multi Router Traffic Grapher)** je voľne dosupný nástroj na monitorovanie sieťového zaťaženia. MRTG generuje HTML stránky obsahujúce PNG a GIF obrázky, ktoré poskytujú živú vizuálnu interpretáciu tejto prevádzky. Pôvodne bolo MRTG vyvinuté Tobiasom Oetikerom a Daveom Random na monitorovanie vyťaženia routrov, neskôr sa rozvinulo na nástroj, ktorý vie vytvárať grafy a štatistiky takmer pre všetko. MRTG používa SNMP na posielanie požiadaviek s dvoma identifikátormi objektov na zariadenie (object identifiers (OIDs)). Zariadenie, ktoré musí mať zapnutý SNMP, bude mať management information base (MIB) pomocou ktorého nájde OID. Po zozbieraní informácií zariadenie pošle surové dáta zabalené v SNMP. MRTG tieto dáta nahrá do logu na klientovi popri minulých dátach, čím vytvorí graf.

## 2.2. Porovnanie sieťových manažment systémov

V tejto kapitole rozoberáme výhody a nevýhody použitia vyššie uvedených monitorovacích systémov. **Výhodou Nagios-u** je možnosť použitia veľkého množstva doplnkov, a teda monitorovania rôznych typov zariadení. Ďalšími výhodami je jednoduchý webový prístup a nadefinovanie závislostí. V prípade výpadku nadradeného zariadenia (napr. sieťového prepínača) nedochádza k oznámeniam o nedostupnosti podradených zariadení. Nagios je vydaný pod licenciou GNU-GPL a v súčasne dobe existuje veľká užívateľská a programátorská komunita, ktorá napomáha rýchlemu riešeniu nedostatkov a vzniknutých chýb v systéme. Na druhej stane **nevýhodou** tohto monitorovacieho systému je zložitá textová konfigurácia, ktorá si vyžaduje skúsenejšieho administrátora sieťových zariadení a nemožnosť konfigurácie pluginov z tretích strán. Nevýhodou tohto systému je aj možnosť použitia SNMP protokolu, ktorého použitie je závislé na použití doplnku do systému.

Ďalším monitorovacím systémom v poradí je **Cacti. Výhodou** tohto systému oproti predchádzajúcemu systému je integrácia použitia SNMP protokolu priamo v monitorovacom systéme. Ďalšími výhodami je veľmi prehľadne spracované a intuitívne webové administračné rozhranie, možnosť sofistikovaného použitia reštrikcií a grafov. **Nevýhodou** tohto systému je nestabilita systému pri spracovaní

grafov (v niektorých prípadoch systém prestane pracovať na niekoľko sekúnd) a dlhá frekvencia obnovy, ktorú nie je možné nastaviť (rádovo niekoľko minút). Aj napriek skutočnosti, že protokol SNMP je priamo implementovaný v systéme a nie je potrebný žiaden doplnok, táto implementácia v niektorých prípadoch nefunguje správne a zobrazuje nesprávne hodnoty.

Tretím hodnoteným systémom bol monitorovací systém **Zabbix**, ktorého hlavnou **výhodou** je veľmi dobrá implementácia SNMP protokolu. Tento systém má podobné monitorovanie ako Nagios a vykresľuje podobné grafy ako Cacti. Výhodou tohto systému je aj intuitívna konfigurácia, prehľadnosť správ a upozornení a rýchle webové administračné rozhranie. Na druhej strane **nevýhodou** je webové administračné rozhranie, ktoré je neintuitívne a oproti textovej konfigurácii sa v ňom ťažko orientuje. Aj keď je výhodou systému intuitívna konfigurácia, proces konfigurácia je zdĺhavý.

Posledným hodnotením monitorovacím systémom je **MRTG.** Jeho **výhodou** je veľká možnosť nastavení a parametrov. Cieľom a súčasne výhodou je pomerne rýchle vykresľovanie rôznych grafov. Tieto grafy zachytávajú najmä sieťovú prevádzku, ktorá môže byť monitorovaná kdekoľvek a kedykoľvek. **Nevýhodou** systému je možnosť straty v dôsledku kompresie údajov. Nevýhodou je tiež problematická inštalácia najnovšie vydaných verzií týchto systémov, premenné musia mať rovnaké škály a limity. Mrtg nie je schopné zaznamenávať údaje z väčšieho množstva sieťových zariadení (približne vyše 600).

# 3. Bezpečnostné aspekty monitorovania sieťových prepínačov

Súčasťou tejto kapitoly je popis tých funkcionalít sieťových prepínačov, ktorých hodnoty je možné monitorovať a ktorých monitorovanie má význam z pohľadu bezpečnosti počítačovej siete.

## 3.1. Sieťový prepínač v rámci ISO/OSI modelu

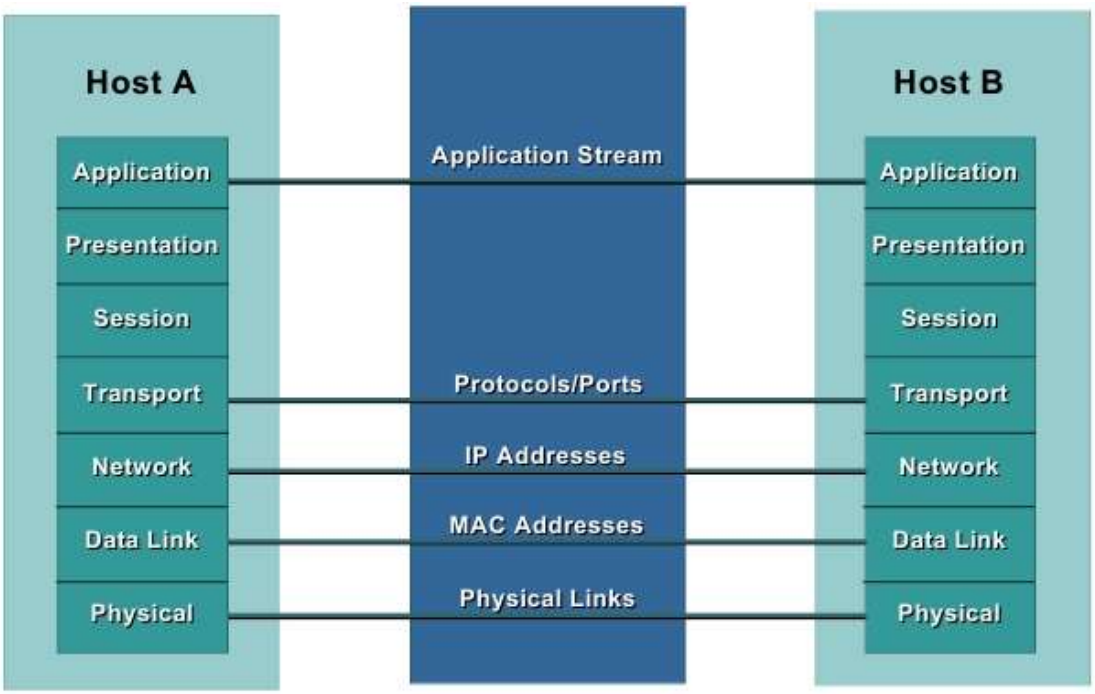
**Sieťový prepínač** (switch) je aktívny prvok prepájajúci jednotlivé segmenty siete. Sieťový prepínač obsahuje menšie alebo väčšie množstvo portov, od dvoch až po niekoľko stoviek, na ktoré sa pripájajú rôzne zariadenia alebo časti siete (podsiete). Najčastejšie sa stretneme s ethernetovým sieťovým prepínačom, ktorý je realizovaný krútenou dvojlinkou. Nahradil staršie zariadenia, huby, ktoré signál len kopírovali do všetkých rozhraní (portov). Okrem vyššieho výkonu (stanice navzájom nesúťažia o signál) je najdôležitejšie to, že signál už nie je zdieľaný a informácie sa posielajú len konkrétnemu adresátovi, tým pádom bezpečnosť je tu oveľa vyššia. Adresovanie sieťového prepínača sa určuje automaticky, učením. Používa sa na to Backward Learning Algorithm, podľa ktorého si sieťový prepínač automaticky vyplní tabuľku identifikujúcu cieľové rozhrania pre jednotlivé adresy.

**Referenčný model OSI (Open Systems Interconnection Reference Model, ISO-OSI)** model sa skladá zo siedmych vrstiev (fyzická, spojová, sieťová, transportná, relačná, prezentačná, aplikačná). Každá z nich vykonáva skupinu jasne definovaných funkcií potrebných pre komunikáciu. Na svoju činnosť využíva služby svojej susednej nižšej vrstvy atď. Podľa referenčného modelu nie je dovolené vynechať nejakú vrstvu, ale niektorá vrstva nemusí byť aktívna. Vtedy sa nazýva nulová alebo transparentná. **Komunikáciu** medzi systémami tvorí :

1) komunikácia medzi vrstvami jedného systému, riadi sa pravidlami, ktoré sa všeobecne nazývajú rozhrania (interface) a
2) komunikácia medzi rovnakými vrstvami rôznych systémov, riadi sa protokolmi.

Na začiatku vznikne požiadavka niektorého z procesov v aplikačnej vrstve. Príslušný podsystém požiada o vytvorenie spojenia prezentačnú vrstvu. V rámci aplikačnej vrstvy je komunikácia s iným systémom riadená aplikačným protokolom.

Podsystémy v prezentačnej vrstve sa dorozumievajú prezentačným protokolom. Takto sa postupuje stále nižšie až k fyzickej vrstve, kde sa použije na spojenie prenosových prostredí. Súčasne sa pri prechode z vyššej vrstvy k nižšej pridávajú k užívateľským (aplikačným) údajom záhlavia jednotlivých vrstiev. Tak dochádza k postupnému zapuzdrovaniu pôvodnej informácie. U príjemcu sa postupne spracovávajú riadiace informácie jednotlivých vrstiev a vykonávajú ich funkcie.




**Obrázok č. 3 – ISO/OSI model**

Pre nás je dôležitá dátová (data lyaer) a sieťová vrstva (network layer). To sú vrstvy, na ktorých pracujú sieťové prepínače. V nasledujúcich kapitolách rozoberieme jednotlivé bezpečnostné aspekty sieťových prepínačov v rámci týchto vrstiev ISO/OSI modelu.

## 3.2. Sieťová prevádzka (network traffic)

**DHCP server spoofing** je jedným z príkladov útokov na sieťovú prevádzku. Je to špeciálny typ útoku, pri ktorom útočník môže získať prístup k sieťovej prevádzke pomocou falšovania odpovedí, ktoré by poslal správny DHCP server. Tento útok používa techniku ARP spoofing, tiež nazývanú ARP poisoning, čo je jednoduchá

technika na útočenie na LAN. ARP spoofing dovolí útočníkovi čítať rámce na LAN sieti, meniť prevádzku, zastaviť prevádzku a tak ďalej.[8]

## 3.3. MAC address table

MAC address flooding alebo MAC address table overflow útoky sú veľmi bežné. Tabuľka MAC adries na sieťovom prepínači obsahuje MAC adresy dostupné na danom fyzickom porte a prislúchajúcu VLAN.

Keď sieťový prepínač príjme rámec, tak sa pozrie do tabuľky MAC adries (tiež nazývanej CAM tabuľka) a hľadá tam cieľovú MAC adresu. Sieťový prepínač používa tabuľku MAC adries na druhej vrstve. Keď rámec príde na port na sieťovom prepínači, zdrojová MAC adresa sa naučí z hlavičky packetu v druhej vrstve a zaznamená sa do tabuľky MAC adries. Keď sa sieťový prepínač už „naučil“ MAC adresu počítača pripojeného na konkrétny port, tak záznam je určite v tabuľke. V tomto prípade sieťový prepínač posiela rámec do konkrétneho počítača na konkrétnom porte. Ak MAC adresa v tabuľke neexistuje, sieťový prepínač sa začne správať ako hub a preposiela každý prijatý rámec na každý port.[9]

## 3.4. Spanning tree protokol

Redundantné linky sú vždy vítané v sieti, pretože zvyšujú jej robustnosť a dostupnosť. Redundantné linky, keď sa na nich pozeráme z hľadiska druhej vrstvy, môžu spôsobiť slučky. Je to preto, lebo TTL (Time To Live) pole packetu je až v hlavičke na tretej vrstve. To znamená, že TTL bude odhalené, až keď bude prechádzať routrom. Tým pádom neexistuje spôsob, ako „zabiť“ packet, ktorý sa zasekol v slučke na druhej vrstve. Táto situácia môže prerásť do broadcastovej búrky. A práve na prevenciu pred týmto stavom slúži spanning tree protocol. Spanning tree protokol dosahuje túto bezslučkovú prevádzku siete pomocou toho, že si určí jeden sieťový prepínač ako hlavný most. Útočník, ktorý musí byť pripojený do lokálnej siete, môže toto nastavenie zmeniť, ak je port, na ktorom je pripojený, v móde trunk. Tým pádom sa útočník môže dostať k údajom, ku ktorým by bežne prístup nemal a taktiež rýchlosť takejto siete sa výrazne spomalí.[10]

## 3.5. Address resolution protokol (ARP)

Klasický man-in-the-middle útok je založený na manipulácii ARP cache lokálnych počítačov a smerovačov. Prebieha to nasledovne. Útočník pošle Gratitious ARPs (GARPs) napríklad nevyžiadané ARP odpovede, ktoré sa spájajú s IP adresou niektorého cieľového počítača a útočníkovou MAC adresou. Následne všetky pakety budú forvardované k útočníkovi. Keď je sieťový prepínač nakonfigurovaný na Dynamic ARP Inspection, tak si sám kontroluje, či IP adresa a MAC adresa spolu naozaj patria.

## 3.6. Bezpečnosť portov sieťového prepínača

Manažovateľné sieťové prepínače majú samozrejme manažovateľné porty. Ak takýto port má nastavenia príliš zhovievavé nastavenia, tak je sieťový prepínač náchylný na útok. Pri nastavovaní portov je potrebné si dať pozor na niekoľko vecí:

- sieťový prepínač by nemal používať defaultné používateľské účty, pretože útočník môže jednoducho tieto účty odhaliť,
- správne nastavené heslo, tzn. silné heslo, aby nemohol byť použitý slovníkový útok; Na rôznych sieťový prepínačoch rôzne heslá, aby keď sa útočník dostane do jedného zariadenia, aby sa automaticky nedostal do ďalších,
- port security - nastavuje sa preto, aby sme predišli arp útokom (nastaví sa počet MAC adries, ktoré sa na sieťový prepínač môžu pripojiť).

## 3.7. Bezpečnosť CAM tabuľky

Základnou myšlienkou port security je predchádzať CAM-table overflow útokov. V tomto útoku by útočník poslal tisíce fiktívnych rámcov s náhodnou MAC adresou až kým CAM-tabel sieťového prepínača nie je plná a sieťový prepínač začne zapĺňať každý podrámec. Potom sa sieťový prepínač začne chovať ako HUB a útočník je schopný falšovať každý rámec. Takýto typ útoku ovplyvňuje len konkrétny VLAN na ktorom je útočník pripojený.

## 3.8. Bezpečnosť VLAN trunk

VLAN hopping útok je možné dosiahnuť dvoma rôznymi spôsobmi, switch spoofingom a Double taggingom.

VLAN hopping útok dovoľuje rámcom z jedného VLANu prechod do iného VLANu bez toho, aby išli cez router. Útočník môže vytvoriť a poslať rámce z jedného VLANu so spoofovaným 802.1G tagom, takže rámce skončia v úplne inom VLANe.

Double tagging funguje podobne. Útočník, ktorý má napríklad VLAN 10, pošle rámce, ktoré sú dvakrát taggované, ako keby bol použitý 802.1Q tag.[11]

## 3.9. Bezpečnosť Cisco discovery protokol

Cisco discovery protocol (CDP) je protokol, ktorý môžu defaultne používať všetky Cisco zariadenia. CDP objavuje ďalšie Cisco zariadenia, ktoré sú priamo napojené. To dovoľuje zariadeniam automaticky si nastaviť pripojenie v niektorých prípadoch. CDP správy nie sú kryptované. V CDP správach sa nachádzajú informácie ako IP adresa, verzia softvéru, platforma, možnosti a VLAN. Útočník môže tieto informácie jednoducho zistiť pomocou WireSharku alebo iného network analyzéra.[12]

## 3.10. Prístup k prepínaču

K sieťovému prepínaču sa dá pristupovať rôzne, napríklad aj cez sieťový (COM) port priamo na ňom. Druhý, častejší typ prístupu je vzdialený.

- SSH
- Telnet
- Web

Najčastejším typom prístupu je telnet, preto na neho existuje najviac útokov. Na protokole telnet je najhoršia jeho bezpečnosť. Všetka komunikácia, ktorú cez telnet posielame, je nekryptovaná, to znamená že je to obyčajný text. Takže akýkoľvek rámec posielame, vystavujeme sa riziku jeho sniffovania. Preto sa v otvorených sieťach teraz štandardne používa SSH. Telnet protokol môže byť použitý na získanie vzdialeného prístupu k CISCO sieťovým prepínačom, ale aj ostatným výrobcom. Keď nastavíme login heslo pre vty riadky stále nemáme dostatočné zabezpečenie. Momentálne

neexistuje spoľahlivá metóda, ktorá by zabránila útočníkovi použiť brute force na získanie hesla. DoS (Denial of Service) je ďalší typ telnet útoku. Útočník pošle veľké množstvo neužitočných a nepodstatných údajových rámcov a zahltí pripojenie.

# 4. Implementácia monitorovacieho systému sieťových prepínačov

Posledná kapitola tejto záverečnej práce sa venuje praktickej implementácii monitorovacieho systému sieťových prepínačov v prostredí lokálnej počítačovej siete. K naplneniu tohto cieľa práce sme vytvorili jednoduchú aplikáciu pre operačný systém Android, ktorá využitím protokolu SNMP získava údaje o sieťových prepínačoch z hľadiska bezpečnosti. Teoretickým základom tejto kapitoly je kapitola venujúca sa protokolu SNMP a kapitola zaoberajúca sa bezpečnostnými aspektmi monitorovania sieťových prepínačov. Na otestovanie jednotlivých funkcionalít systému sme použili sieťové prepínače od spoločnosti Cisco – Catalyst 2960.

## 4.1. Analýza systému

### 4.1.1. Požiadavky

Systém vznikol v dôsledku potreby monitorovania sieťových prepínačov v rámci počítačovej siete ŠDaJ UPJŠ. Napriek už existujúcim manažment systémom v rámci univerzitnej počítačovej siete (systémy NAGIOS, MRTG) vznikla potreba získavať základné informácie o sieťových prepínačoch v akomkoľvek čase a na ktoromkoľvek mieste, kde je umožnený prístup k Internetu. Cieľom systému by nemalo byť získanie všetkých dostupných informácii, ale len základných a najčastejšie požadovaných, ktorých znalosť môže pomôcť pri prvotnej reakcii na vzniknutý problém.

Medzi základné požiadavky kladené na tento systém zaraďujeme:

- možnosť sledovania základných informácií o sieťovom prepínači,
- možnosť sledovania bezpečnostných informácií o sieťovom prepínači,
- sledovanie z akéhokoľvek zariadenia a
- intuitívne a jednoduché grafické rozhranie.

### 4.1.2. Návrh systému

Pred tvorbou samotného systému bolo potrebné si stanoviť niekoľko základných okruhov problémov, ktoré bolo potrebné vyriešiť. Išlo o tieto problémy:

- účel systému,
- náčrt spôsobu fungovania systému a
- výber vhodného protokolu.

**Účelom systému** bude umožniť správcom sieťových prepínačov monitorovanie týchto zariadení z hľadiska bezpečnosti. Ako už bolo spomenuté v predchádzajúcej kapitole, cieľom systému bude získavanie len základných informácií. Výber, aké informácie je potrebné získavať, sme spravili vzhľadom na tretiu kapitolu tejto práce s názvom bezpečnostné aspekty monitorovania sieťových prepínačov.

Druhý okruh problémov je **spôsob fungovania systému**. Systém by mal umožňovať administrátorovi sieťových prepínačov si možnosť zvoliť všeobecné a určité bezpečnostné informácie, ktoré chce získať o danom sieťovom prepínači a po zadaní určitých parametrov by mal dostať potrebné informácie.

Posledným okruhom problémov bol výber vhodného protokolu. Bolo potrebné vybrať taký protokol, ktorý by umožňoval jednoduchý prístup k sieťovému prepínaču a veľkú rozšíriteľnosť medzi sieťovými prepínačmi. Najvhodnejším kandidátom bol SNMP protokol, ktorý je najrozšírenejším protokol v rámci manažmentu počítačových sietí.

Po výbere tohto protokolu bolo potrebné sa rozhodnúť, ktorú časť protokolu použijeme. Najprv sme uvažovali o využití **SNMP TRAP** – čo sú bezpečnostné správy posielané od agenta na managera. Ide o silnú pomôcku pri správe počítačovej siete. Keďže jedna z požiadaviek systému je možnosť monitorovania sieťových prepínačov z akéhokoľvek zariadenia s operačným systémom Android, nebolo možné použiť SNMP TRAP, keďže tie sú zasielané na konkrétnu IP adresu. V tomto prípade by sme museli buď zabezpečiť preposielanie správ zo zariadenia s touto IP adresou, alebo zabezpečiť prideľovanie konkrétnej IP adresy nášmu zariadeniu. Z týchto dôvodov sme sa rozhodli použiť **SNMP GET** a získavať údaje zo sieťového prepínača podľa vopred určených OID.

## 4.2. Prostriedky použité pri vývoji systému

Celú aplikáciu sme vyvíjali pomocou programovacieho jazyka JAVA v programovacom prostredí Eclipse. Keďže ide o aplikáciu pre operačný systém Android, pri vývoji aplikácie sme použili Android SDK. V nasledujúcich kapitolách podrobnejšie rozoberieme jednotlivé prostriedky použité pri vývoji systému.

### 4.2.1. Eclipse a Android Devolopment tools

**Eclipse** je programovacie prostredie pre vývoj viacerých programovacích jazykov napísané z väčšej časti v jave. Používa sa najmä na vývoj aplikácií v programovacom jazyku Java. Použitím rôznych doplnkov je možné toto prostredie použiť aj pre iné programovacie jazyky. Príkladom takéhoto doplnku je **Android Development Tools** (ADT), ktorý umožňuje vývoj aplikácií pre operačný systém Android. ADT rozširuje možnosti Eclipsu, vďaka čomu môžeme rýchlejšie vyvíjať Android projekty, vytvárať grafické aplikácie, pridávať balíčky založené na Android Framework API.

**Android** sme si ako vývojové prostredie zvolili z niekoľkých dôvodov:

- pre Android sa ako programovací jazyk používa JAVA,
- Android je open-source platforma
- zastúpenie Androidu na trhu je markantné,
- knižnica, ktorá implementuje SNMP protokol je na Android v rámci open-source licencie a
- aplikácie pre Android je možné inštalovať z viacerých zdrojov.

### 4.2.2. Android emulátor

Za účelom testovania aplikácie slúži Android emulátor. Je to plne funkčná náhrada zariadenia s operačným systémom Android, ktoré sa spúšťa na počítači. Jeho nevýhodou je spúšťacia doba, ktorá je asi 2 minúty. Ale zároveň je možné zapnúť viacero inštancií naraz, ktoré spolu vedia komunikovať, napríklad cez bluetooth. Jeho

ďalšou výhodou je LogCat. Ide o textový výstup každej operácie vykonanej na emulátore, vrátane chýb, čo je z hľadiska vývoja a testovania systému veľmi užitočné.

### 4.2.3. Protokol SNMP

Protokol SNMP sme venovali samostatnú kapitolu. Tiež sme sa ním okrajovo zaoberali pri sieťových manažment systémoch a ich porovnaní. Pri implementácii systému sme použili SNMP protokol vo verzii 2c. Keďže ide len o monitoring a nie celý manažment sieťových prepínačov, používame len požiadavku GET a odpoveď na ňu.

### 4.2.4. Programovací jazyk Java

Pre samotný vývoj systému sme použili programovací jazyk **Java.** Tento jazyk, ktorý bol vyvíjaný **Sun Microsystems** a bol vydaný v roku 1995, patrí v dnešnej dobe ku skupine najviac používaných programovacích jazykov. Java má podobnú syntax ako programovacie jazyky C a C++. Aplikácie vyvinuté pomocou tohto progamovacieho jazyka môžu bežať na akomkoľvek Java virtuálnom stroji (**Java Virtual Machine, JVM**), čo má za následok jeho rozšíriteľnosť a použiteľnosť na všetkých platformách operačných systémov. Na druhej strane to spôsobuje jeho pomalosť oproti iným rozšíreným programovacím jazykom.

### 4.2.5. Knižnica SNMP4J

Pre implementáciu SNMP protokolu v rámci programovacieho jazyka Java sme zvolili knižnicu SNMP4J, ktorá je open-source knižnicou pre programovací jazyk Java. Podporuje vytváranie príkazov na strane managera v rámci SNMP protokolu, ako aj vytváranie odpovedí na strane agenta. Vývoj tejto knižnice bol inšpirovaný SNMP++ knižnicou, ktorá sa používa pre programovací jazyk C++. Medzi základné výhody SNMP4J radíme [12]:

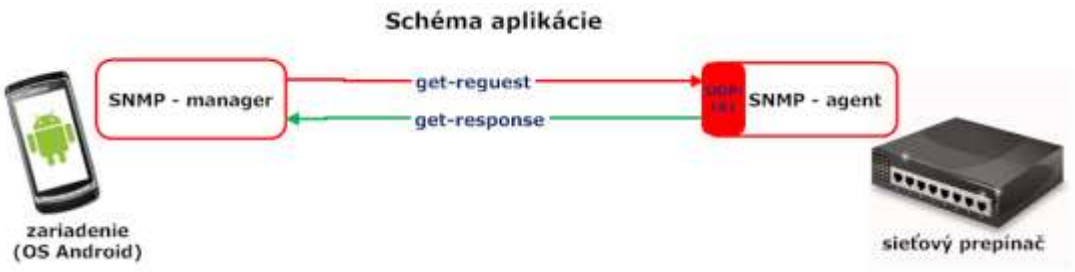
- podpora SNMPv3 s použitím MD5 and SHA a DES, 3DES, AES 128, AES 192, and AES 256 bezpečnosťou,
- všetky PDU typy,
- synchrónne a asynchrónne požiadavky,

- podporu generátora príkazov a odpovedí na nich,
- open-source s Apache licenciou,
- podporu Java™ 1.4.1 alebo vyššej verzie,
- logovanie založené na Log4J,
- multi-threading suport,
- implementáciu pre SNMP-TARGET-MIB, SNMP-NOTIFICATION-MIB, SNMP-PROXY-MIB, SNMP-FRAMEWORK-MIB, SNMPv2-MIB, SNMP-COMMUNITY-MIB, SNMP-USER-BASED-SM-MIB, SNMP-VIEW-BASED-ACM-MIB, NOTIFICATION_LOG-MIB, a SNMP-MPD-MIB, SNMP4J-PROXY-MIB, SNMP-TLS-TM-MIB, SNMP-TSM-MIB,
- podporu pre IPv4/IPv6 UDP, TCP, a TLS

## 4.3. Popis implementácie

### 4.3.1. Základný popis systému

Systém využíva SNMP protokol, ktorému sme sa bližšie venovali v kapitole 3.1. tejto práce. Protokol SNMP používa pre komunikáciu dve časti – managera a agenta. Nami vytvorený systém je vlastne manager, ktorý spracúva vstupnú požiadavku od administrátora pre zistenie konkrétnych informácii. Túto požiadavku prevedenie na konkrétne OID a pošle svoju požiadavku na agenta – **get-request** na sieťový port 161/UDP. Tento agent vytiahne podľa požadovaného oid hodnotu z MIB tabuľky sieťového prepínača a pošle túto hodnotu na managera – **get-response**. Systém spracuje odpoveď a vypíše ju na výsledkovej obrazovke.




**Obrázok č. 4 – Grafické rozhranie aplikácie – úvodná obrazovka**

Na tomto mieste by sme chceli upozorniť na problém MIB tabuliek a ich OID. Pri vývoji a testovaní systému sa stávalo, že nebolo možné získať potrebné údaje z mnohých vybraných OID napriek tomu, že MIB tabuľky, ktoré tieto záznamy obsahujú, sú na danom type prepínača implementované.

Systém sa skladá z troch tried, ktoré rozširujú triedu Activity:

- Pripojenie
- Zadanie požiadaviek
- Výpis

### 4.3.2. Trieda pripojenie

Hlavnou funkciou tejto triedy je získavať od používateľa základné informácie, ktoré sú potrebné k pripojeniu sa k sieťovému prepínaču pomocou SNMP protokolu. Ide o **IP adresu** (vo verzii 4) a **community name string**, čo sa rieši v metóde **onClick**, ktorá je priradená tlačidlu „OK“. V ostatných metódach implementujeme grafické rozhranie prvého z troch layoutov, a to pripojenie.xml. Úplný výpis triedy pripojenie uvádzame v prílohe č. A – Trieda pripojenie.

Informácie požadované od sieťového prepínača sú delené na dve skupiny:

- **základné informácie a**
- **bezpečnostné informácie.**

Medzi základné informácie zaraďujeme napríklad čas behu zariadenia, popis zariadenia, správca zariadenia, meno zariadenia, teplota zariadenia, lokalita zariadenia, stav pamäte a procesora zariadenia. Medzi bezpečnostné informácie radíme status vlanov, status portov, cam tabuľka a pod.

### 4.3.3. Trieda zadanie požiadaviek

Podstatou tejto triedy je získavanie informácií, na ktoré chce používateľ získať odpovede od sieťového prepínača. „Otázka“, ktorú užívateľ odosiela na sieťový prepínač je realizovaná pomocou OID, ktoré sme získavali z **MIB tabuľky** daného sieťového prepínača. Príkladom OID je meno sieťového prepínača. Jeho OID záznam

je ".1.3.6.1.2.1.1.5.0". Parametre sa nastavujú pomocou checkbox komponentov. Ako layout tu používame zadaniepoziadaviek.xml. Úplný výpis triedy pripojenie uvádzame v prílohe č. B – Trieda zadanie požiadaviek.

### 4.3.4. Trieda výpis

V predošlých triedach sme informácie získavali. Účelom tejto triedy je poskytovať základné informácie. Celé získavanie prebieha v metóde **vypisVysledok** pomocou SNMP verzie 2c. Na ukážku prikladám aj XML súbor s layoutom vypis.xml, ktorý som použil v tejto triede. Úplný výpis triedy pripojenie uvádzame v prílohe č. C – Trieda výpis.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        <TextView
            android:id="@+id/textViewVypis"
            android:layout_width="fill_parent"
            android:layout_height="480dp"
            android:text=" "
            android:textAppearance="?android:attr/textAppearanceLarge" />
    </ScrollView>
</LinearLayout>
```

### 4.3.5. Grafické prostredie

Grafické prostredie aplikácie bolo vytvorené prostredníctvom android layoutu, ktorý sa realizuje pomocou xml súborov.

Systém obsahuje 3 grafické obrazovky:

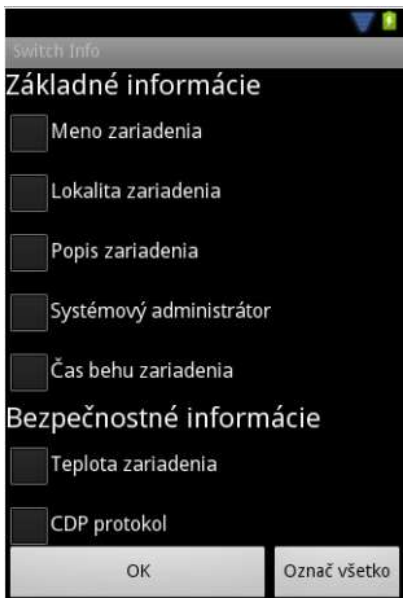
- úvodná obrazovka
- konfiguračná obrazovka
- výsledková obrazovka

Súčasťou **úvodnej obrazovky** sú dve polia, do ktorých sa zadáva IP adresa sieťového prepínača a public community name, ktoré je nastavené na sieťovom

prepínači minimálne v režime read-only (len na čítanie). Súčasťou sú aj tri tlačidlá, a to potvrdenie údajov a prechod na ďalšiu obrazovku, vyčistenie údajov a ukončenie systému. Úvodnú obrazovku môžeme vidieť na obrázku č. 5


**Obrázok č. 5 – Grafické rozhranie aplikácie – úvodná obrazovka**

Druhou obrazovkou je **konfiguračná obrazovka**. Tá umožňuje používateľovi systému zaškrtnúť údaje, ktoré si chce načítať zo sieťového prepínača. Tieto údaje sú rozdelené do dvoch častí – základné informácie a bezpečnostné informácie. Konfiguračnú obrazovku môžeme vidieť na obrázku č. 6




**Obrázok č. 6 – Grafické rozhranie aplikácie – konfiguračná obrazovka**

Poslednou obrazovkou systému je **výsledková obrazovka.** Na tejto obrazovke je možné vidieť požadované informácie o danom sieťovom prepínači.

# Záver

V práci sa venujeme monitorovaniu lokálnej počítačovej siete. Rozoberáme jednotlivé protokoly a prostriedky pre monitorovanie tejto siete. Zameriavame sa najmä SNMP protokolu, ktorý je najpoužívanejším protokolom pre monitoring, resp. pre celý manažment počítačových sietí. V ďalšej časti práce sme sa zaoberali sieťovým manažment systémom. Keďže existuje veľké množstvo takýchto systémov, vybrali sme len štyroch zástupcov. Kritériami pre ich výber bola open-source licencia a spolupráca so SNMP protokolom. Výsledok porovnávania uvádzame v kapitole 2.2. Podľa nášho porovnania každý z týchto systémov má svoje výhody a aj nevýhody. Pre skúsených administrátorov by sme odporúčali systém Zabbix, pre začínajúcich správcov systém Cacti. Nasledujúca časť práce rozoberala bezpečnostné aspekty monitorovania sieťových prepínačov. Definovali sme pojem sieťový prepínač a načrtli jeho postavenie v rámci referenčného OSI modelu. Uviedli sme deväť bezpečnostných aspektov dotýkajúcich sa sieťových prepínačov. Tieto poznatky sme neskôr využili pri návrhu a vývoji monitorovacieho systému Switch info. Vychádzajúc zo získaných poznatkov sme naprogramovali aplikáciu pre operačný systém Android - Switch Info. Pre jej vývoj sme využili programovací JAVA a SNMP4J knižnicu. Bližšie sa touto knižnicou zaoberáme v kapitole 4.2.5. Vývoj aplikácie spočíval vo vyhľadávaní, selektovaní a analyzovaní dostupných existujúcich riešení. Následne bolo potrebné získať materiály a nájsť dokumentáciu k MIB tabuľkám jednotlivých sieťových prepínačov. Zistili sme, že implementácie jednotlivých MIB tabuliek v rámci konkrétnych sieťových prepínačov je problematická. Často sa stávalo, že záznamy OID nefungovali a nebolo možné získať požadované údaje o sieťovom prepínači. Ďalším krokom vývoja aplikácie bola príprava bezpečného testovacieho prostredia (sieťové prepínače CISCO Catalyst 2960, CISCO Catalyst 3560, HP ProCurve 2650, HP ProCurve 2626 ) a jeho konfigurácia podľa nami stanovených požiadaviek. Nasledujúcim krokom vo vývoji aplikácie bolo samotné programovanie, ktorého výsledkom je aplikácia Switch Info. Táto aplikácia v jednoduchom užívateľskom rozhraní poskytuje možnosť základnej komunikácie so sieťovým prepínačom prostredníctvom protokolu SNMPv2c. V budúcnosti by sme chceli pokračovať vo vývoji aplikácie Switch Info. Implementovať do nej ďalšie metódy, pomocou ktorých by sme dokázali informácie zo sieťových prepínačov nielen získavať, ale ich aj nastavovať. To by znamenalo nielen jeho monitorovanie, ale aj jeho

manažment. Pri implementácii by sme chceli využiť vyššiu verziu protokolu SNMP - SNMP v.3, keďže v tejto práci sme využili SNMP vo verzii 2c.

# Zoznam použitej literatúry


[1] CLEMM, A.: Network Management Fundamentals. Cisco Press, 2006. 552 s.

[2] LISKA, A.: The Practice of Network Security: Deployment Strategies for Production Environments. New Jersey : Prentice Hall Professional, 2003. 391 s.

[3] RFC 1189 – The Common Management Information Services and Protocols for the Internet (CMOT and CMIP) [online]. [cit. 2012-05-24]. Dostupné z www: http://tools.ietf.org/html/rfc1189.

[4] Bejtlich, R.: The Tao of Network Security Monitoring - Beyond Intrusion Detection. Addison-Wesley, 2004, 798 s.

[5] RFC 1028 - A Simple Gateway Monitoring Protocol [online]. [cit. 2012-05-27]. Dostupné z www: http://tools.ietf.org/html/rfc1028.

[6] RFC 1156 - Management Information Base for Network Management of TCP/IP-based internets [online]. [cit. 2012-05-28]. Dostupné z www: http://tools.ietf.org/html/rfc1156.

[7] RFC 1213 - Management Information Base for Network Management of TCP/IP-based internets: MIB-II [online]. [cit. 2012-05-28]. Dostupné z www: http://tools.ietf.org/html/rfc1213.

[8] VALTER, P.: How does the internet work - Spoofing Attacks – DHCP Server Spoofing [online]. 2011, [cit. 2012-06-03]. Dostupné z www: http://howdoesinternetwork.com/2011/spoofing-attack/.

[9] VALTER, P.: How does the internet work - Spoofing Attacks – MAC Address Flooding – MAC address table overflow attacks [online]. 2011, [cit. 2012-06-03]. Dostupné z www: http://howdoesinternetwork.com/2011/mac-address-flooding/

[10] VALTER, P.: How does the internet work - Spoofing Attacks – STP Layer 2 attack – Manipulating Spanning Tree Protocol settings [online]. 2011, [cit. 2012-06-04]. Dostupné z www: http://howdoesinternetwork.com/2012/stp-attack/

[11] VALTER, P.: How does the internet work - Spoofing Attacks – VLAN hopping attack – Switch Spoofing and Double tagging [online]. 2011, [cit. 2012-06-02]. Dostupné z www: http://howdoesinternetwork.com/2012/vlan-hopping-attack/

[12] VALTER, P.: How does the internet work - Spoofing Attacks – CDP Attacks – Cisco Discovery Protocol Attack [online]. 2011, [cit. 2012-06-02]. Dostupné z www: http://howdoesinternetwork.com/2011/cdp-attack/

[13] SNMP4J [online]. 2011, [cit. 2012-06-02]. Dostupné z www: http://www.snmp4j.org/.

# Prílohy

Príloha A: Trieda pripojenie

Príloha B: Trieda zadanie požiadaviek

Príloha C: Trieda výpis

Príloha D: CD médium – zdrojový kód k aplikácii Switch Info

# Príloha A

```
package get.from.oid;
import org.snmp4j.CommunityTarget;
import org.snmp4j.PDU;
import org.snmp4j.Snmp;
import org.snmp4j.TransportMapping;
import org.snmp4j.event.ResponseEvent;
import org.snmp4j.mp.SnmpConstants;
import org.snmp4j.smi.Address;
import org.snmp4j.smi.OID;
import org.snmp4j.smi.OctetString;
import org.snmp4j.smi.UdpAddress;
import org.snmp4j.smi.VariableBinding;
import org.snmp4j.transport.DefaultUdpTransportMapping;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
public class Vypis extends Activity {
	String IP;
	String komunita;
	String vybratePoziadavky;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.vypis);
		TextView vypis = (TextView) findViewById(R.id.textViewVypis);
		vybratePoziadavky = getIntent().getExtras().getString("vybrate");
		komunita = getIntent().getExtras().getString("komunita");
		IP = getIntent().getExtras().getString("IP");
		if (!vybratePoziadavky.equals("")) {
		 if (vybratePoziadavky.contains("Meno zariadenia")) {
			vypis.setText(vypis.getText()+"Meno zariadenia: \n\n
"+Meno_zariadenia()+"\n\n\n");
```

```
                }
                if (vybratePoziadavky.contains("Popis zariadenia")) {
                        vypis.setText(vypis.getText() + "Popis zariadenia: \n\n " +
Popis_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Systémový administrátor")) {
                        vypis.setText(vypis.getText() + "Systémový administrátor: \n\n " +
System_administrator()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Čas behu zariadenia")) {
                        vypis.setText(vypis.getText() + "Čas behu zariadenia: \n\n " +
Cas_behu_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Teplota zariadenia")) {
                        vypis.setText(vypis.getText() + "Teplota zariadenia: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Lokalita zariadenia")) {
                        vypis.setText(vypis.getText() + "Lokalita zariadenia: \n\n " +
Lokalita_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Stav pamate")) {
                        vypis.setText(vypis.getText() + "Stav pamate: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Využitie procesora")) {
                        vypis.setText(vypis.getText() + "Využitie procesora: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Vlan status")) {
                        vypis.setText(vypis.getText() + "Vlan status: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("CDP protokol")) {
                        vypis.setText(vypis.getText() + "CDP protokol: \n\n " +
Teplota_zariadenia()+"\n\n\n");
```

```
                }
                if (vybratePoziadavky.contains("Používatelia")) {
                        vypis.setText(vypis.getText() + "Používatelia: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("CAM tabuľka")) {
                        vypis.setText(vypis.getText() + "CAM tabuľka: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Spanning tree protokol")) {
                        vypis.setText(vypis.getText() + "Spanning tree protokol: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("DHCP server")) {
                        vypis.setText(vypis.getText() + "DHCP server: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("SSH status")) {
                        vypis.setText(vypis.getText() + "SSH status: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                if (vybratePoziadavky.contains("Telnet status")) {
                        vypis.setText(vypis.getText() + "Telnet status: \n\n " +
Teplota_zariadenia()+"\n\n\n");
                }
                }
        }
        public String Cas_behu_zariadenia() {
                String oid = "1.3.6.1.2.1.1.3.0";
                return snmpGet(IP, komunita, oid);
        }
        public String Popis_zariadenia() {
                String oid = ".1.3.6.1.2.1.1.1.0";
                return snmpGet(IP, komunita, oid);
        }
```

```
public String System_administrator() {
        String oid = ".1.3.6.1.2.1.1.4.0";
        return snmpGet(IP, komunita, oid);
}
public String Meno_zariadenia() {
        String oid = ".1.3.6.1.2.1.1.5.0";
        return snmpGet(IP, komunita, oid);
}
public String Teplota_zariadenia() {
        String oid = ".1.3.6.1.2.1.1.5.0";
        return snmpGet(IP, komunita, oid);
}
public String Lokalita_zariadenia() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Stav_pamata() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Vyuzitie_procesora() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Vlan_status() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Cdp_protokol() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Pouzivatelia() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
```

```
}
public String Cam_tabulka() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Spanning_tree_protocol() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Dhcp_server() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Ssh_status() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String Telnet_status() {
        String oid = ".1.3.6.1.2.1.1.6.0";
        return snmpGet(IP, komunita, oid);
}
public String snmpGet(String address, String komunita, String OID) {
        String str = "";
        try {
         OctetString komunita2 = new OctetString(komunita);
         address = address + "/" + 161;
         Address cielovaAdresa = new UdpAddress(address);
         TransportMapping transport = new DefaultUdpTransportMapping();
         transport.listen();
         CommunityTarget comm = new CommunityTarget();
         comm.setCommunity(komunita2);
         comm.setVersion(SnmpConstants.version2c);
         comm.setAddress(cielovaAdresa);
         comm.setRetries(2);
         comm.setTimeout(5000);
```

```
                PDU pdu = new PDU();
                ResponseEvent response;
                Snmp snmp;
                pdu.add(new VariableBinding(new OID(OID)));
                pdu.setType(PDU.GET);
                snmp = new Snmp(transport);
                response = snmp.get(pdu, comm);
                if (response != null) {
                        if (response.getResponse().getErrorStatusText()
                                        .equalsIgnoreCase("Success")) {
                                PDU pduresponse = response.getResponse();
                                str = pduresponse.getVariableBindings().firstElement()
                                                .toString();
                                if (str.contains("=")) {
                                        int len = str.indexOf("=");
                                        str = str.substring(len + 1, str.length());}
                        }
                } else {
                        System.out.println("Time out");
                }
                snmp.close();
                } catch (Exception e) {
                e.printStackTrace();
                }
                Log.v("skuska", str);
                return str;
        }
}
```

# Príloha B

```
package get.from.oid;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
public class ZadaniePoziadaviek extends Activity {
	public boolean onCreateOptionsMenu(Menu menu) {
		super.onCreateOptionsMenu(menu);
		menu.add("Select all");
		menu.add("Deselect all");
		return true;
	}
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.zadaniepoziadaviek);
		Button OK = (Button) findViewById(R.id.OK_zadanie_button);
		OK.setOnClickListener(new View.OnClickListener() {
		@Override
		public void onClick(View arg0) {
			String IP = getIntent().getExtras().getString("IP");
			String komunita = getIntent().getExtras().getString("komunita");
			Intent vypis = new Intent("android.intent.action.VYPISANIE");
			vypis.putExtra("IP", IP);
			vypis.putExtra("komunita", komunita);
			CheckBox Meno_zariadenia = (CheckBox)
findViewById(R.id.Meno_zariadenia);
			CheckBox Lokalita_zariadenia = (CheckBox)
findViewById(R.id.Lokalita_zariadenia);
			CheckBox Popis_zariadenia = (CheckBox)
findViewById(R.id.popis_zariadenia);
```

```
CheckBox System_administrator = (CheckBox) findViewById(R.id.System_administrator);
CheckBox Cas_behu_zariadenia = (CheckBox) findViewById(R.id.Cas_behu_zariadenia);
CheckBox Teplota_zariadenia = (CheckBox) findViewById(R.id.Teplota_zariadenia);
CheckBox Stav_pamata = (CheckBox) findViewById(R.id.Stav_pamate);
CheckBox Vyuzitie_procesora = (CheckBox) findViewById(R.id.Vyuzitie_procesora);
CheckBox Vlan_status = (CheckBox) findViewById(R.id.VLAN_status);
CheckBox Cdp_protokol = (CheckBox) findViewById(R.id.CDP_protokol);
CheckBox Pouzivatelia = (CheckBox) findViewById(R.id.Pouzivatelia);
CheckBox Cam_tabulka = (CheckBox) findViewById(R.id.CAM_tabulka);
CheckBox Spanning_tree_protocol = (CheckBox) findViewById(R.id.Spanning_tree_protokol);
CheckBox Dhcp_server = (CheckBox) findViewById(R.id.DHCP_server);
CheckBox Ssh_status = (CheckBox) findViewById(R.id.SSH_status);
CheckBox Telnet_status = (CheckBox) findViewById(R.id.Telnet_status);
String vybrate = "";
if (Meno_zariadenia.isChecked()) {
        vybrate += "Meno zariadenia ";
}
if (Lokalita_zariadenia.isChecked()) {
        vybrate += "Lokalita zariadenia ";
}
if (Popis_zariadenia.isChecked()) {
        vybrate += "Popis zariadenia ";
}
if (System_administrator.isChecked()) {
        vybrate += "Systémový administrátor ";
}
if (Cas_behu_zariadenia.isChecked()) {
        vybrate += "Čas behu zariadenia ";
}
if (Teplota_zariadenia.isChecked()) {
```

```
                vybrate += "Teplota zariadenia ";
            }
            if (Stav_pamata.isChecked()) {
                vybrate += "Stav pamate ";
            }
            if (Vyuzitie_procesora.isChecked()) {
                vybrate += "Využitie procesora ";
            }
            if (Vlan_status.isChecked()) {
                vybrate += "Vlan status ";
            }
            if (Cdp_protokol.isChecked()) {
                vybrate += "CDP protokol ";
            }
            if (Pouzivatelia.isChecked()) {
                vybrate += "Používatelia ";
            }
            if (Cam_tabulka.isChecked()) {
                vybrate += "CAM tabuľka ";
            }
            if (Spanning_tree_protocol.isChecked()) {
                vybrate += "Spanning tree protokol ";
            }
            if (Dhcp_server.isChecked()) {
                vybrate += "DHCP server ";
            }
            if (Ssh_status.isChecked()) {
                vybrate += "SSH status ";
            }
            if (Telnet_status.isChecked()) {
                vybrate += "Telnet status ";
            }
            vypis.putExtra("vybrate", vybrate);
            startActivity(vypis);
        }
```

```
        });
        final Button Oznac = (Button) findViewById(R.id.select_all_button);
        Oznac.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final CheckBox Meno_zariadenia = (CheckBox) findViewById(R.id.Meno_zariadenia);
            CheckBox Lokalita_zariadenia = (CheckBox) findViewById(R.id.Lokalita_zariadenia);
            CheckBox Popis_zariadenia = (CheckBox) findViewById(R.id.popis_zariadenia);
            CheckBox System_administrator = (CheckBox) findViewById(R.id.System_administrator);
            CheckBox Cas_behu_zariadenia = (CheckBox) findViewById(R.id.Cas_behu_zariadenia);
            CheckBox Teplota_zariadenia = (CheckBox) findViewById(R.id.Teplota_zariadenia);
            CheckBox Stav_pamata = (CheckBox) findViewById(R.id.Stav_pamate);
            CheckBox Vyuzitie_procesora = (CheckBox) findViewById(R.id.Vyuzitie_procesora);
            CheckBox Vlan_status = (CheckBox) findViewById(R.id.VLAN_status);
            CheckBox Cdp_protokol = (CheckBox) findViewById(R.id.CDP_protokol);
            CheckBox Pouzivatelia = (CheckBox) findViewById(R.id.Pouzivatelia);
            CheckBox Cam_tabulka = (CheckBox) findViewById(R.id.CAM_tabulka);
            CheckBox Spanning_tree_protocol = (CheckBox) findViewById(R.id.Spanning_tree_protokol);
            CheckBox Dhcp_server = (CheckBox) findViewById(R.id.DHCP_server);
            CheckBox Ssh_status = (CheckBox) findViewById(R.id.SSH_status);
            CheckBox Telnet_status = (CheckBox) findViewById(R.id.Telnet_status);
            Meno_zariadenia.setChecked(true);
            Lokalita_zariadenia.setChecked(true);
            Popis_zariadenia.setChecked(true);
            System_administrator.setChecked(true);
            Cas_behu_zariadenia.setChecked(true);
            Teplota_zariadenia.setChecked(true);
            Stav_pamata.setChecked(true);
```

```
                    Vyuzitie_procesora.setChecked(true);
                    Vlan_status.setChecked(true);
                    Cdp_protokol.setChecked(true);
                    Pouzivatelia.setChecked(true);
                    Cam_tabulka.setChecked(true);
                    Spanning_tree_protocol.setChecked(true);
                    Dhcp_server.setChecked(true);
                    Ssh_status.setChecked(true);
                    Telnet_status.setChecked(true);
                }
                });
            }
}
```

# Príloha C

```
package get.from.oid;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class Pripojenie extends Activity {
	EditText ipAdresa;
	EditText publicCommunity;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.pripojenie);

		Button gombik = (Button) findViewById(R.id.exit_app_button);
		gombik.setOnClickListener(new View.OnClickListener() {
		public void onClick(View view) {
			finish();
		}
		});

		Button getInfo = (Button) findViewById(R.id.ok_info_button);
		ipAdresa = (EditText) findViewById(R.id.IP_switch);
		publicCommunity = (EditText) findViewById(R.id.Read_Community);

		getInfo.setOnClickListener(new View.OnClickListener() {
		@Override
		public void onClick(View arg0) {
			Intent poziadavky = new Intent("android.intent.action.POZIADAVKY");

			poziadavky.putExtra("IP", ipAdresa.getText().toString());
```

```
                poziadavky.putExtra("komunita",        publicCommunity.getText().toString());


                startActivity(poziadavky);
        }
        });

        Button clear = (Button) findViewById(R.id.clear_button);
        clear.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
                ipAdresa = (EditText) findViewById(R.id.IP_switch);
                publicCommunity = (EditText) findViewById(R.id.Read_Community);
                ipAdresa.setText("");
                publicCommunity.setText("");
        }
        });
}
```