SNMP (Simple Network Management Protocol)

1.Visão geral:

O SNMP é um protocolo desenvolvido para gerência de rede, através de objetos ocorrências podem ser contabilizadas.

A contabilização do SNMP pode ser realizada de forma direta ou indireta.

A forma direta é quando o sistema operacional ou software manipula diretamente a variável, a forma indireta ocorre quando a variável é manipulada indiretamente através de uma outra referência.

Para exemplificar o modo direto podemos imaginar a contagem de frames ou datagramas que entram ou saem via interface do equipamento em estudo.

Esse exemplo foi retirado do Xinu[1] o arquivo net_in.c:

```
#include <conf.h>
#include <kernel.h>
#include <network.h>
#include <ospf.h>
int arp_in(struct netif *, struct ep *);
int rarp_in(struct netif *, struct ep *);
int ip_in(struct netif *, struct ep *);
 /*------------------------------------------------------------------------
* ni_in - network interface input function
*------------------------------------------------------------------------ */
int ni_in(struct netif *pni, struct ep *pep, unsigned len) {
  int rv;
  pep-&gt;
  ep_ifn = pni - &amp;
  nif[0];
  /* record originating intf # */
 pni-&gt;ni_ioctets += len;
 if (!memcmp(pni-&gt;ni_hwa.ha_addr, pep-&gt;
    ep_dst, EP_ALEN)) pni-&gt;
    ni_iucast++;
 else pni-&gt;
    ni_inucast++;
switch (pep-&gt;ep_type) {
    case EPT_ARP: rv = arp_in(pni, pep);
     break;
case EPT_RARP: rv = rarp_in(pni, pep);
     break;
case EPT_IP: rv = ip_in(pni, pep);
     break;
default: pni-&gt;
    ni_iunkproto++;
freebuf(pep); rv = OK;
}
   return rv;
}
```

Nesse caso as variáveis “ni_iunkproto” e “ni_iucast” contam quantos frames chegam destinados para protocolos desconhecidos e de unicast respectivamente.

Para acompanhar melhor, logo abaixo estão as variáveis usadas pelo driver da placa de rede no Xinux[1] na struct “netif” na sessão “MIB”:

```
/* Interface MIB */
        char        *ni_descr;      /* text description of hardware */
        int         ni_mtype;       /* MIB interface type           */
        long        ni_speed;       /* bits per second              */
        char        ni_admstate;    /* administrative status (NIS_*)*/
        long        ni_lastchange;  /* last state change (1/100 sec)*/
        long        ni_ioctets;     /* # of octets received         */
        long        ni_iucast;      /* # of unicast received        */
        long        ni_inucast;     /* # of non-unicast received    */
        long        ni_idiscard;    /* # dropped - output queue full*/
        long        ni_ierrors;     /* # input packet errors        */
        long        ni_iunkproto;   /* # in packets for unk. protos */
        long        ni_ooctets;     /* # of octets sent             */
        long        ni_oucast;      /* # of unicast sent            */
        long        ni_onucast;     /* # of non-unicast sent        */
        long        ni_odiscard;    /* # output packets discarded   */
        long        ni_oerrors;     /* # output packet errors       */
        long        ni_oqlen;       /* output queue length          */
        long        ni_maxreasm;    /* max datagram can reassemble  */
```

Para exemplificar a forma indireta podemos imaginar quando temos que informar o tempo que o sistema está ativo, ele subtrai a data e horário atual daquela que foi armazenado quando o sistema foi iniciado.

Para ilustrar melhor foi coletado um exemplo no Linux no arquivo “/proc/net/snmp”, nesse arquivo são colocadas informações sobre os frames, datagramas e pacotes que são tratados pelo kernel.

```
bash# cat /proc/net/snmp

Ip: Forwarding DefaultTTL InReceives InHdrErrors InAddrErrors ForwDatagrams
InUnknownProtos InDiscards InDelivers OutRequests OutDiscards OutNoRoutes
ReasmTimeout ReasmReqds ReasmOKs ReasmFails FragOKs FragFails
FragCreates
Ip: 1 64  38  0 0 0 0 0  20 38 0 0 0 0 0 0 0 0
Icmp: InMsgs InErrors InDestUnreachs InTimeExcds InParmProbs InSrcQuenchs
InRedirects InEchos InEchoReps InTimestamps InTimestampReps InAddrMasks
InAddrMaskReps OutMsgs OutErrors OutDestUnreachs OutTimeExcds
OutParmProbs OutSrcQuenchs OutRedirects OutEchos OutEchoReps
OutTimestamps OutTimestampReps OutAddrMasks OutAddrMaskReps
Icmp: 20 0 18 0 0 0 0  1 1  0 0 0 0  19 0 18 0 0 0 0 0  1  0 0 0 0 0
Tcp: RtoAlgorithm RtoMin RtoMax MaxConn ActiveOpens PassiveOpens
AttemptFails EstabResets CurrEstab InSegs OutSegs RetransSegs InErrs OutRsts
Tcp: 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Udp: InDatagrams NoPorts InErrors OutDatagrams
Udp: 0 18 0 18
```

1.1– Principais características do SNMP:

- Gerenciamento de redes;
- Controlar equipamentos de rede;
- Detectar problemas e erros;
- Usa o paradigma busca/armazenamento;
- Porta 161/UDP.

2. Versões do SNMP:

- **SNMP versão 1:**

A recomendação existente para a versão um do SNMP é a RFC 1157, suas principais vantagens são a simplicidade, flexibilidade e popularidade.

Mas, a versão um possui algumas desvantagens como pouca preocupação com a segurança, com a mensagem SNMP é enviada a senha sem criptografia no campo “community”.

Para o acesso de busca costuma -se ter a senha “public” e para acesso de armazenamento costuma -se usar a senha “private”.

- **SNMP versão 2:**

Nessa versão existe maior preocupação com segurança na transmissão de senhas.

Foram adicionadas mensagens como: “GetBulkRequest” e “InformRequest/Response”.

A recomendação existente para a versão 1 do SNMP é a RFC 1448.

- **SNMP versão 3:**

Esse trouxe maior avanço em segurança.

O desenvolvimento dessa versão suporta algoritmos criptográficos na de senhas.

O único problema é que muitos equipamentos não suportam essa versão, sendo necessário realizar atualizações.

A recomendação existente para a versão 1 do SNMP é a RFC 3414.

3. Vocabulário usado:

- **Gerente:**
  Aquele que guarda as informações e realiza pedidos no agente.

- **Agente:**
  Aquele que recebe a requisição do gerente e coleta os dados na “MIB”.

- **MIB (Management Information Base):**
  E coletânea dos objetos que se que gerenciar, de forma estruturada em uma árvore hierárquica.

- **ASN.1 (Abstract Syntax Natition):**
  Determina formas como as variáveis serão representadas.
  Garante interoperabilidade entre arquiteturas diferentes, evitando erros de representações.

- **SMI (Structure Management information):**
  Determina regras de como a MIB será estruturada (RFC 1155, RFC

1212, RFC 1215). Define tipo de dados e regras e atributos de nomes.

Para exemplificar melhor, logo abaixo está o arquivo “asn1.h” do Xinux[1]:

```
/* asn1.h - A1_SIGNED */

/* constants for parsing an SNMP packet, according to ASN.1                    */
/* ASN.1 object types */
#define ASN1_SEQ    0x30        /* sequence object                  */
#define ASN1_INT    0x02        /* integer                          */
#define ASN1_OCTSTR         0x04        /* octet string                                    */
#define ASN1_NULL 0x05          /* null                                            */
#define ASN1_OBJID0x06          /* object identifier                */
#define ASN1_IPADDR         0x40        /* ip address                                      */
#define ASN1_COUNTER        0x41        /* counter                                         */
#define ASN1_GAUGE          0x42        /* gauge                                           */
#define ASN1_TIMETICKS      0x43        /* time ticks                                      */
/* Protocol Data Unit types -- SNMP specific                                    */
#define PDU_GET             0xA0        /* get request                                     */
#define PDU_GETN    0xA1        /* get-next request                 */
#define PDU_RESP    0xA2        /* response                         */
#define PDU_SET             0xA3        /* set request                                     */
#define PDU_TRAP    0xA4        /* trap message                     */
/* Constants used for conversion of objects to/from ASN.1 notation  */
#define CHAR_BITS  8                    /* number of bits per char          */
#define CHAR_HIBIT          0x80                /* octet with the high bit set         */
#define BYTE2_HIBIT         0x8000              /* 2 bytes with high bit set           */
#define BYTE3_HIBIT         0x800000  /* 3 bytes with high bit set          */
#define BYTE4_HIBIT         0x80000000          /* 4 bytes with high bit set           */
#define     A1_SIGNED(x)        ((x) == ASN1_INT)
/* the standard MIB prefix - 1.3.6.1.2.1                                        */
extern char MIB_PREFIX[];
/* the standard MIB prefix is encoded by ASN.1 into 5 octets                    */
#define MIB_PREF_SZ         5
```

Podemos ver os códigos usados para representar tipos de dados (ASN1_INT, ASN1_IPADDR e ASN1_COUNTER) e comandos (PDU_GET, PDU_SET e PDU_TRAP).

- **Objetos:**

  É a abstração daquilo que se quer monitorar. Pode ser na forma de variáveis ou estruturas.

- **CMIP (Common Management Information Protocol):**

  Esse protocolo segue o padrão ISO-OSI e trabalha na camada de transporte.

  Muito usado em telecomunicações principalmente em telefonia, ele é muito complexo comparado ao SNMP.

3. Solicitações enviadas:

O SNMP versão 1 trabalha com os comandos:

- “Get”: Para realizar buscas nas MIBs.
- “Set”: Para armazenamento nas MIBs.
- “Trap”: Para informar erros ou alguma anomalia.

Os comandos usados podem ser visto no arquivo asn1.h do xinu[1] mostrado logo acima na sessão “vocabulário usado” no item “SMI”.

4. Tipos de MIB:

- **MIB I:**
  Foi a primeira MIB desenvolvida (RFC1066), simples .

- **MIB II:**
  Uma evolução da MIB I (RFC 1213).
  Nessa foram adicionados diversos objetos.

- **MIB Experimental:**
  Define as MIBs que ainda estão em desenvolvimento.

- **MIB Privada:**
  Define a MIB desenvolvida por fabricantes que atendem a características particulares dos equipamentos.

5. Aspecto da MIB:

Arquivo “mib.h” do Xinu[1]:

```
/* mib information structure: 1 per mib entry;  Initialized in snmib.c  */
struct mib_info {
        char    *mi_name;       /* name of mib variable in English      */
        char    *mi_prefix;      /* prefix in English (e.g., "tcp.")     */
        struct  oid mi_objid;   /* object identifier                    */
        int     mi_vartype;       /* type: integer, aggregate, octet str  */
        Bool    mi_writable;   /* is this variable writable?           */
        Bool    mi_varleaf;     /* is this a leaf with a single value   */
        int     (*mi_func)(struct snbentry *, struct mib_info *, int);
                                  /* function to implement get/set/getnext*/
        char    *mi_param;      /* parameter used with function         */
        struct  mib_info *mi_next;  /* pointer to next var. in lexi-order*/
};
```

Essa estrutura usado no XINU[1] mostra como a MIB é manipulada.

- **mi_name:** Temos o nome do objeto.

- **mi_objid:** Temos a localização desse objeto na árvore, tendo como referência a posição da MIB.

Segue o padrão ASN.1, codifica os nós em vez de nome por números. Exemplo: “osi.org.dod” será “1.3.6” .

- **mi_vartype:** Tipo da variável definida em “asn1.h” pelo padrão ASN.1 (mib.h do XINU[1]).

- **mi_writable:** Define permissão de escrita (mib.h do XINU[1]).

- **mi_varleaf:** Define se esse ponto é um folha da árvore, ou seja, se esse tem um valor ou tem um filho (mib.h do XINU[1]).

- **mi_func:** Função que irá tratar esse objeto ou manipular dados realizando conversões.

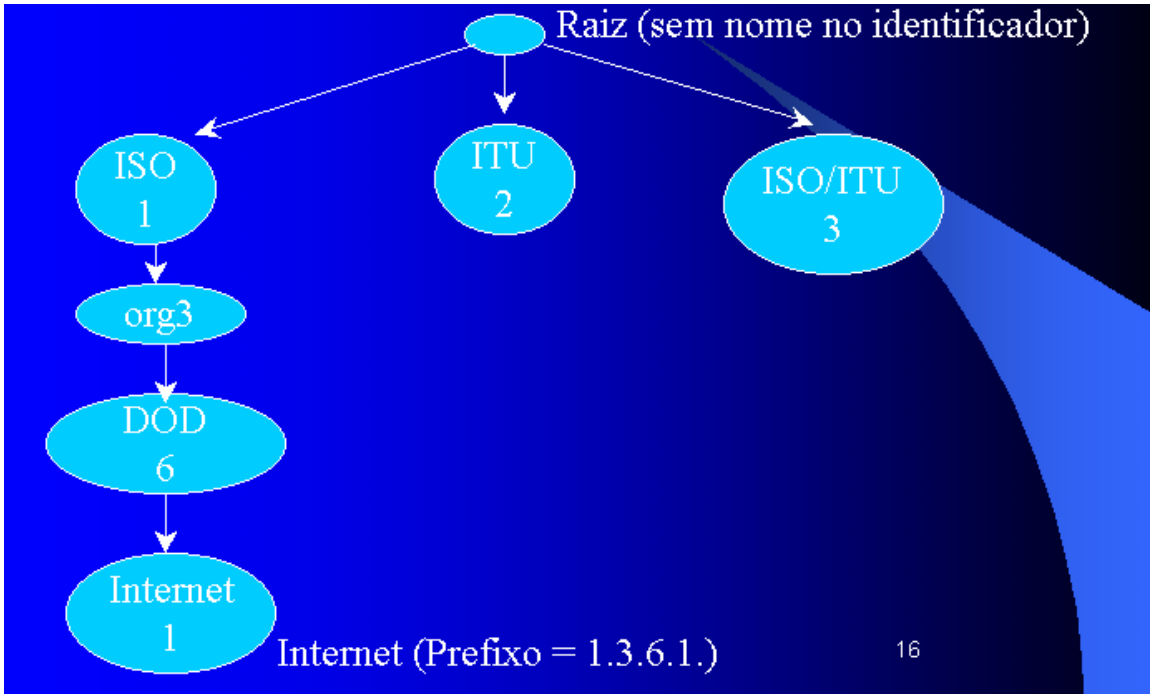
- **mi_next:** Ponteiro para o próximo objeto lexiográfico, lembrando que a árvore é organizada por números e pontos, pontos esses separando os níveis para facilitar a busca lexiográfica. O ponteiro “mi_next” é usado pelo comando “Get Next” para localizar o próximo item.

6. MIB:

A MIB é organizada em uma árvore, cada nó recebe um nome hierárquico como identificador.

Quando a estrutura interna manipula a árvore são dados números para facilitar a busca dos objetos.

No exemplo abaixo o nó “osi.org.dod” pode usar o prefixo “1.3.6”, lembrar que a raiz não é nomeada.

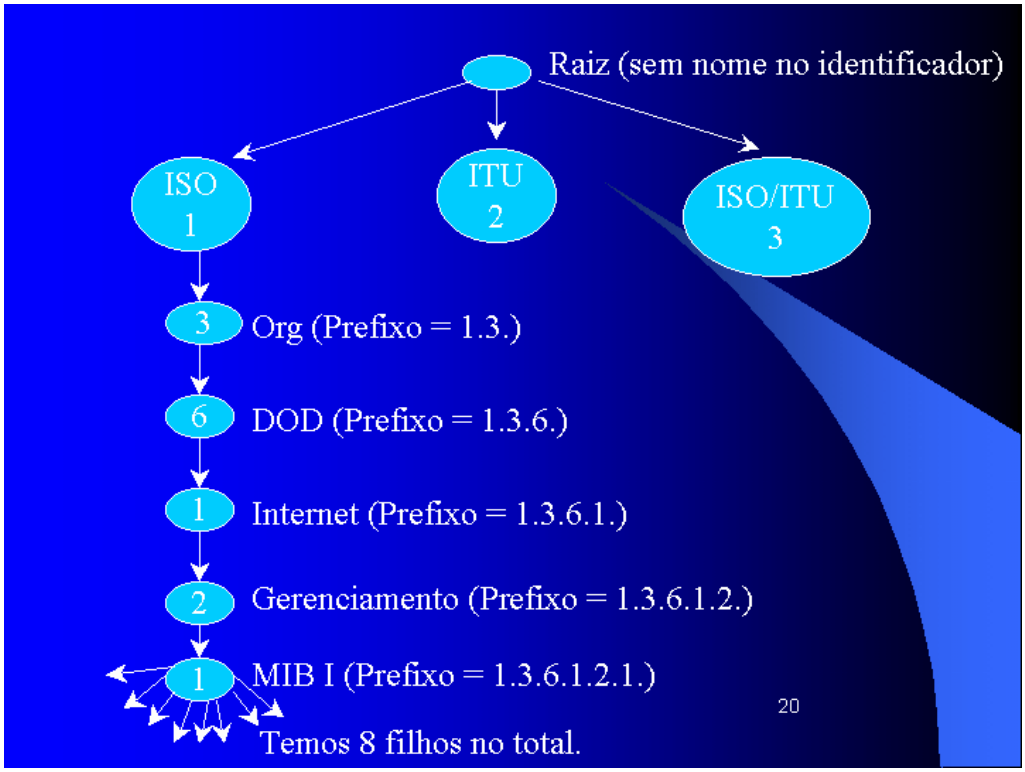


Os identificadores usados podem ser visto no arquivo “snmvars.h” do XINU[1]:

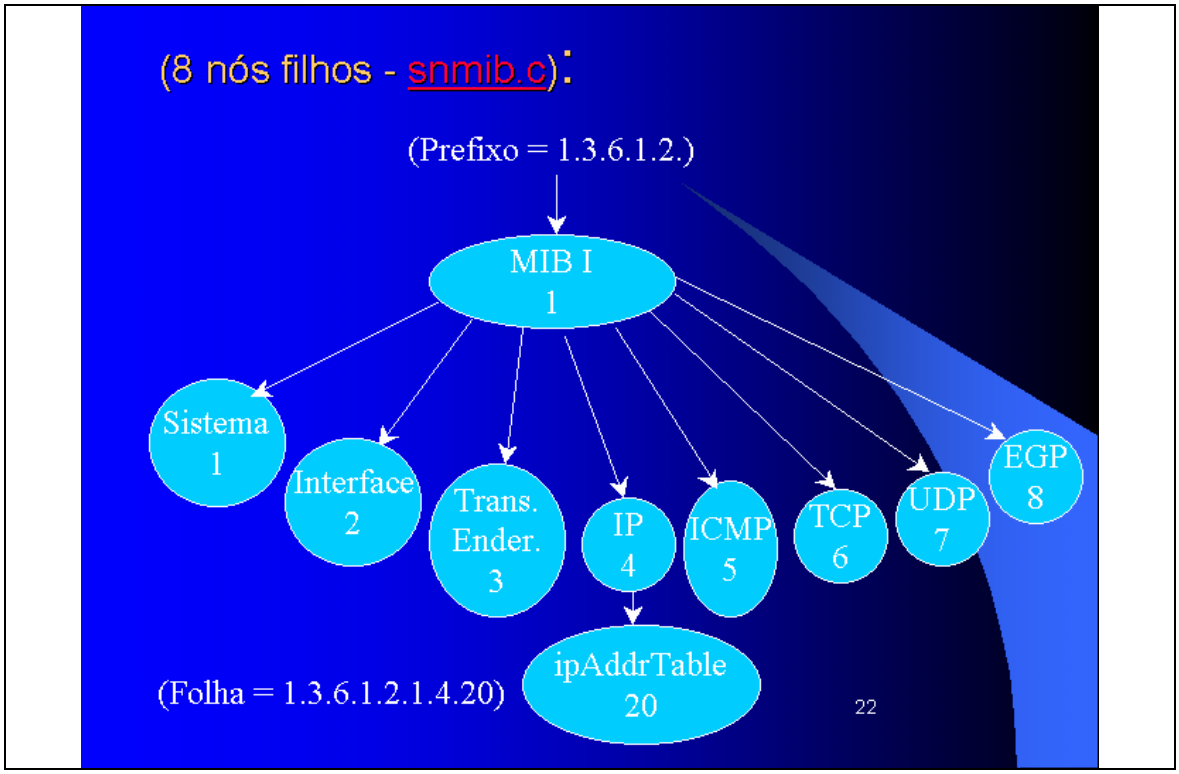
```
/* snmpvars.h */
/* System & Interface MIB */
extern char		SysDescr[], SysContact[], SysName[], SysLocation[];
extern unsigned
		SysUpTime, SysServices, IfNumber;
/* IP MIB */
extern unsigned
		IpForwarding, IpDefaultTTL, IpInReceives, IpInHdrErrors,
		IpInAddrErrors, IpForwDatagrams, IpInUnknownProtos, IpInDiscards,
		IpInDelivers, IpOutRequests, IpOutDiscards, IpOutNoRoutes,
		IpReasmTimeout, IpReasmReqds, IpReasmOKs, IpReasmFails, IpFragOKs,
		IpFragFails, IpFragCreates, IpRoutingDiscards;
/* ICMP MIB */
extern unsigned
		IcmpInMsgs, IcmpInErrors, IcmpInDestUnreachs, IcmpInTimeExcds,
		IcmpInParmProbs, IcmpInSrcQuenchs, IcmpInRedirects, IcmpInEchos,
```

```
            IcmpInEchoReps, IcmpInTimestamps, IcmpInTimestampReps,
            IcmpInAddrMasks, IcmpInAddrMaskReps, IcmpOutMsgs, IcmpOutErrors,
            IcmpOutDestUnreachs, IcmpOutTimeExcds, IcmpOutParmProbs,
            IcmpOutSrcQuenchs, IcmpOutRedirects, IcmpOutEchos,
            IcmpOutEchoReps, IcmpOutTimestamps, IcmpOutTimestampReps,
            IcmpOutAddrMasks, IcmpOutAddrMaskReps;
/* UDP MIB */
extern unsigned
            UdpInDatagrams, UdpNoPorts, UdpInErrors, UdpOutDatagrams;
/* TCP MIB */
extern unsigned
            TcpRtoAlgorithm, TcpRtoMin, TcpRtoMax, TcpMaxConn, TcpActiveOpens,
            TcpPassiveOpens, TcpAttemptFails, TcpEstabResets, TcpCurrEstab,
            TcpInSegs, TcpOutSegs, TcpRetransSegs, TcpInErrs, TcpOutRsts;
/* SNMP MIB */
extern unsigned
      snmpInPkts, snmpOutPkts, snmpInBadVersions,
      snmpInBadCommunityNames, snmpInBadCommunityUses,
      snmpInASNParseErrs, snmpInTooBigs, snmpInNoSuchNames,
      snmpInBadValues, snmpInReadOnlys, snmpInGenErrs,
      snmpInTotalReqVars, snmpInTotalSetVars, snmpInGetRequests,
      snmpInGetNexts, snmpInSetRequests, snmpInGetResponses,
      snmpInTraps, snmpOutTooBigs, snmpOutNoSuchNames,
      snmpOutBadValues, snmpOutGenErrs, snmpOutGetRequests,
      snmpOutGetNexts, snmpOutSetRequests, snmpOutGetResponses,
      snmpOutTraps, snmpEnableAuthenTraps;
```

Um exemplo da MIB até o nó “MIB” alvo do nosso estudo:



Já abaixo do nó “MIB” temos 8 nós filhos (sistema, Interface, Transformação de Endereço/Valor, IP, ICMP, TCP, UDP e EGP):



Podemos ver nesse exemplo a folha ipAddrTable que tem como identificador “1.3.6.1.2.1.4.20”,

7. ASN.1:

Uma das principais funções do ASN.1 é a representação numérica da árvore da MIB.

Os números são escritos em 7 bits, o oitavo bit mais significativo é usado para representar continuidade da representação no octeto seguinte ou término.

Ou seja, números maiores que 127 o bit mais significativo tem valor 1 e números menores e iguais à 127 o oitavo bit tem valor zero.

Abaixo temos um exemplo retirado do livro “Interligação em rede com TCP/IP – Douglas E. Comer – Volume 1 – Capítulo 21”.

Nesse exemplo o número "909904" maior que 127 é separado em grupos de 7 bits, o oitavo bit é colocado como um para indicar continuidade.

Os demais números menores e iguais a 127 o oitavo bit é colocado como zero, pois, esses não precisão de outro octeto para representa-los.

Se essa representação não fosse usada um nó somente poderia ter 255 nós filhos usando a representação de números.

8. Softwares SNMP:

- Proprietários:

  SunNet Manager, OpenView e MibBrowser.

- GPL:

  Scotty, Snmp-Perl, CMU-snmp e MRTG, Net-SNMP.

- Cracker:

  ADMsnmp

9. Usando o software Net-SNMP e ADMsnmp:

Para exemplificar o uso do SNMP usaremos o Net-SNMP (http://www.net-snmp.org/).

O Net-SNMP é composto de diversos programas, mostraremos os principais.

Para localizar um equipamento com SNMP ativo basta usar o programa de varredura de portas "nmap":

```
$ nmap –p 161,162  -sU  10.10.10.0/24
```

9.1 – O "snmpwalk" realiza busca com o comando Get-Next em equipamentos com suporte a SNMP:

```
$ snmpwalk  XyZ.xxx.yyy.zzz -c public

SNMPv2-MIB::sysDescr.0 = STRING: Bridge in 7600 Card Rev 1.0: ZipChip Rev 2.0, i960,
Nucleus OS, Software Rev Ver.  4.62
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.43.1.8.16
SNMPv2-MIB::sysUpTime.0 = Timeticks: (8039557) 22:19:55.57
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING:
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 2
IF-MIB::ifNumber.0 = INTEGER: 5
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifIndex.4 = INTEGER: 4
IF-MIB::ifIndex.5 = INTEGER: 5
IF-MIB::ifDescr.1 = STRING: Fast Ethernet port # 3.3
IF-MIB::ifDescr.2 = STRING: Fast Ethernet port # 3.4
IF-MIB::ifDescr.3 = STRING: Fast Ethernet port # 3.11
IF-MIB::ifDescr.4 = STRING: Fast Ethernet port # 3.12
IF-MIB::ifDescr.5 = STRING: 7600 Card Rev 1.0 Emulated LAN port
IF-MIB::ifType.1 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.3 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.4 = INTEGER: ethernetCsmacd(6)
..
.. <continuando>
..
```

Como podemos ver a listagem são os nós da árvore da MIB do equipamento alvo.

9.2 - O "snmpnetstat" pode ser usado para obter várias informações

como rotas, portas e serviço ativos, contabilidade de protocolos e outras.

No exemplo abaixo temos a listagem das rotas.

```
$ snmpnetstat -r Xyz.xxx.yyy.www  public|more
Warning: positional community parameter is deprecated. Use -c
Routing tables
Destination          Gateway              Flags   Interface
default              gateway1.xxx.com     UG      if0
x.w.23.10/32         gateway1.xxx.com     UG      le0
yy.xx.yx.203/32      gateway1.xxx.com     UG      le0
zz.wx.172.51/32      gateway1.xxx.com     UG      le0
```

No exemplo abaixo temos a listagem das portas e serviços ativos.

```
$ snmpnetstat -a Xyz.yyy.zzz.www  public
Warning: positional community parameter is deprecated. Use -c
Active Internet (tcp) Connections (including servers)
Proto Local Address          Foreign Address          (state)
tcp   *.*                    *.*                      CLOSED
tcp   *.ssh                  *.*                      LISTEN
tcp   *.smtp                 *.*                      LISTEN
tcp   *.time                 *.*                      LISTEN
tcp   *.sunrpc               *.*                      LISTEN
tcp   *.uucp                 *.*                      LISTEN
tcp   *.submissi             *.*                      LISTEN
tcp   *.4045                 *.*                      LISTEN
tcp   *.5987                 *.*                      LISTEN
tcp   *.x11                  *.*                      LISTEN
tcp   *.32771                *.*                      LISTEN
tcp   *.32777                *.*                      LISTEN
tcp   *.32778                *.*                      LISTEN
tcp   *.32779                *.*                      LISTEN
```

9.3 - O “snmpset” pode alterar os parâmetros na MIB caso tenhamos acesso para armazenamento.

Inicialmente temos que fazer a busca para escolher um nó alvo na MIB:

```
$ snmpwalk  x.y.z.w -c  public

SNMPv2-MIB::sysDescr.0 = STRING: Bridge in 7600 Card Rev 1.0: ZipChip Rev 2.0, i960,
Nucleus OS, Software Rev Ver.  4.62
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.43.1.8.16
SNMPv2-MIB::sysUpTime.0 = Timeticks: (8136406) 22:36:04.06
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING:
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 2
IF-MIB::ifNumber.0 = INTEGER: 5
IF-MIB::ifIndex.1 = INTEGER: 1
```

```
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifIndex.4 = INTEGER: 4
IF-MIB::ifIndex.5 = INTEGER: 5
IF-MIB::ifDescr.1 = STRING: Fast Ethernet port # 3.1
IF-MIB::ifDescr.2 = STRING: Fast Ethernet port # 3.2
IF-MIB::ifDescr.3 = STRING: Fast Ethernet port # 3.9
IF-MIB::ifDescr.4 = STRING: Fast Ethernet port # 3.10
..
.. <continua>
..
```

Nesse caso o alvo será o nó "sysName.0".
Usando o comando “snmpset”:

```
$ snmpset -c private -v 1 x.y.z.w  system.sysName.0 s "teste123"

SNMPv2-MIB::sysName.0 = STRING: teste123
```

Após a alteração podemos ver "sysName.0" modificado:

```
$ snmpwalk  x.y.z.w -c  public
SNMPv2-MIB::sysDescr.0 = STRING: Bridge in 7600 Card Rev 1.0: ZipChip Rev 2.0, i960,
Nucleus OS, Software Rev Ver.  4.62
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.43.1.8.16
SNMPv2-MIB::sysUpTime.0 = Timeticks: (8136406) 22:36:04.06
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: teste123
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 2
IF-MIB::ifNumber.0 = INTEGER: 5
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifIndex.4 = INTEGER: 4
..
.. <continua>
..
```

Um exemplo do uso do comando "getnext" pode ser mostrado com o "snmpgetnext", podemos ver que ele mostra o nó seguinte do “sysName.0” :

```
$ snmpgetnext  x.y.z.w -c public   sysName.0
 SNMPv2-MIB::sysLocation.0 = STRING:
```

9.4 - Um outro software muito usado para quebra de senhas é o ADMsnmp (ftp://ADM.isp.at/ADM/), ele funciona com auxílio de um arquivo de senhas.

Exemplo: Com acesso a MIB com a senha "private".

```
$ ./snmp x.y.z.w -wordfile snmp.passwd
ADMsnmp vbeta 0.1 (c) The ADM crew
ftp://ADM.isp.at/ADM/
greets: !ADM, el8.org, ansia
>>>>>>>>>>> get req name=router  id = 2 >>>>>>>>>>>
>>>>>>>>>>> get req name=cisco  id = 5 >>>>>>>>>>>
>>>>>>>>>>> get req name=public   id = 8 >>>>>>>>>>>
>>>>>>>>>>> get req name=private  id = 11 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 12 name = private ret =0 <<<<<<<<<<
>>>>>>>>>>>> send setrequest id = 12 name = private >>>>>>>>
>>>>>>>>>>> get req name=admin  id = 14 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 13 name = private ret =0 <<<<<<<<<<
>>>>>>>>>>> get req name=proxy  id = 17 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 140 name = private ret =0 <<<<<<<<<<
>>>>>>>>>>> get req name=write  id = 20 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 140 name = private ret =0 <<<<<<<<<<
>>>>>>>>>>> get req name=access  id = 23 >>>>>>>>>>>
>>>>>>>>>>> get req name=root  id = 26 >>>>>>>>>>>
>>>>>>>>>>> get req name=enable  id = 29 >>>>>>>>>>>
>>>>>>>>>>> get req name=all private  id = 32 >>>>>>>>>>>
>>>>>>>>>>> get req name= private  id = 35 >>>>>>>>>>>
>>>>>>>>>>> get req name=test  id = 38 >>>>>>>>>>>
>>>>>>>>>>> get req name=guest  id = 41 >>>>>>>>>>>
<!ADM!>        snmp check on x.y.z.w            <!ADM!>
sys.sysName.0:ADMsnmp
name = private _[1write access
```

10. Conclusão:

Com o crescimento na utilização de equipamentos móveis, redes sem infra-estrutura e equipamentos domésticos em rede os protocolos de gerencia precisaram sofrer simplificações em função da carência de processamento e memória RAM disponíveis nesses equipamentos.

Quando os equipamentos domésticos estiverem sendo usados será necessária uma preocupação com a segurança no que tange: acesso, autenticação e ações de controle nesses equipamentos.

11. Referências:

[1] – Sitema Operacional Xinu comentado pelos livros [2] e [3].

[2] - Livro: "Interligação de rede com TCP/IP Volume 2"
Autor: Douglas E. Commer

[3] - Livro: "Interligação de rede com TCP/IP Volume 1"
Autor: Douglas E. Commer

[4] – Site: GTA/UFRJ – http://www.ufrj.br/