Shteryana Shopova,
syrinx@FreeBSD.org

SNMP Monitoring

BSDCon, Sofia
October, 2007

* the mark FreeBSD is a registered trademark of The FreeBSD Foundation and is used by Shteryana Shopova with the permission of The FreeBSD Foundation

About me

- Computer Science student at University of Sofia, Faculty of Mathematics and Informatics
- Software engineer at Telco Systems, BG Office
- FreeBSD commiter (src) since October, 2006
- Google Summer of Code student (2005 and 2006) and mentor (2007)

What is SNMP?

- Simple Network Monitoring Protocol
- Security Not My Problem
- SNMP version 1 introduced in 1988 (RFC 1157)
- SNMPv2c
  - Community-based SNMP
  - Draft - RFCs 1901-1908
  - De facto standard
- SNMPv3
  - RFCs 3411–3418
  - Finally added authentication, privacy and access control
  - Message encryption with shared key – DES-CBC
  - View-Based Access Control Model (VACM)

SNMP Architecture

- SNMP agents and management stations
- SNMP engines
- MIB (Management Information Base)
- Object definitions via ASN.1 (Abstract Syntax Notation One) encoding
- SMI - Structure of Management Information
    - subset of ASN.1
    - specified in RFCs 2578-2580
    - defines sets of related objects
    - grouped in MIB modules

Monitoring packages available

- A lot of them out there - “#ls -l /usr/ports/net-mgmt/ | wc -l” shows 237
- MRTG - The Multi Router Traffic Grapher, extremely popular
- Nagios (tm) - a lot of features, making it a very powerful monitoring tool
- Zabbix - supports XML data import/export
- However, most GPL-licensed, require X to get the nice manager-friendly plots
- Even more closed source monitoring tools available

Net-SNMP package

- De facto standard Open Source SNMP implementation
- Features SNMP agent, console-based SNMP client tools, snmptrapd
- No X required
- GPL-licensed, supports SNMPv3
- Features a lot of standard MIB implementations
- More details on http://www.net-snmp.org/

bsnmpd(1) - pros and cons

- BSD licensed - code may be used in commercial products
- Already in base system, most bug reporting and all changes are made through the official FreeBSD GNATS system and CVS repository
- Light-weight and easily extensible
- Does not support SNMPv3 (yet)
- Includes modules for monitoring *BSD/FreeBSD specific features such as pf(4) and netgraph(4)

Writing your own modules

- Easy if you are fluent in C coding and are aware of FreeBSD's and (specifically) bsnmpd(1)'s internals, and a SNMP guru
- Google-ing for a patch out there that already does what you need always helps
- A good starter project for (FreeBSD/Networking) enthusiasts (or university students looking for ideas on what to present as a Networking class project)

1) Define a MIB

- You have to be familiar (to some extent) with ASN.1 and SMI
- If a standard MIB is available - better support it
- Example – a module definition and a leaf object definition

```
FOO-MIB DEFINITIONS ::= BEGIN
IMPORTS
   MODULE-IDENTITY
      FROM SNMPv2-SMI;

fooModule MODULE-IDENTITY
    LAST-UPDATED "202001300000Z"
    ORGANIZATION "Foo Org"
    CONTACT-INFO
       "          Your Name
    Postal:    Some Address
    Fax:   +XXX
    E-mail:
       your_email@some_domain.org
       "

DESCRIPTION
"Some description required here."
::= { mgmt 1150 }
END
```

```
fooObject OBJECT-TYPE
     SYNTAX       INTEGER {
               foo1(1),
               foo2(2),
            }
     MAX-ACCESS  read-write
     STATUS      current
     DESCRIPTION
             "Enter some
  description what foo serves
  for here.
             Also maybe some
  description of its possible
  values."
     DEFVAL      { 1 }
     ::= { fooModule 1 }
```

2) Create a .def file

- libsmi(3) – ports/net-mgmt/libsmi/
- gensnmpdef(1) - not compiled and installed with base system but sources available under src/contrib/bsnmp/gensnmpdef
- SMIPATH environment variable
- The contents of the MIB file serve as input for gensnmpdef – typically saved as xxx_tree.def
- The contents of the xxx_tree.def file serve as input for gensnmptree(1) when building the modules
- bsd.snmpmod.mk

3) Makefile - example

```
#
# $FreeBSD: src/usr.sbin/bsnmpd/modules/snmp_bridge/Makefile,v 1.2
  2006/12/07 22:36:17 syrinx Exp $
#

MOD=    bridge
SRCS=   bridge_snmp.c bridge_if.c bridge_port.c bridge_addrs.c \
        bridge_pf.c bridge_sys.c
CFLAGS+= -DSNMPTREE_TYPES

XSYM=   dot1dBridge newRoot topologyChange begemotBridgeNewRoot \
        begemotBridgeTopologyChange begemotBridgeBaseName

MAN=    snmp_bridge.3

BMIBS=  BRIDGE-MIB.txt BEGEMOT-BRIDGE-MIB.txt RSTP-MIB.txt
DEFS=   ${MOD}_tree.def
INCS=   ${MOD}_snmp.h

.include <bsd.snmpmod.mk>
```

4) libbsnmp

- man bsnmplib(3), bsnmpagent(3), bsnmpclient(3), snmpmod(3)
- each module is defined in a struct snmp_module

```
struct snmp_module {
      const char *comment;
      int (*init)(struct lmodule *, int argc, char *argv[]);
      int (*fini)(void);
      void (*idle)(void);
      void (*dump)(void);
      void (*config)(void);
      void (*start)(void);
      proxy_err_f proxy;
      const struct snmp_node *tree;
      u_int tree_size;
      void (*loading)(const struct lmodule *, int);
};
```

5) code the module

- The .def file contains the names of the function that will be called when a GET/SET operation is invoked on the object

```
int
op_dot11StationConfigTable(struct snmp_context *ctx __unused,
     struct snmp_value *val __unused, u_int sub __unused,
     u_int iidx __unused, enum snmp_op op __unused)
{
         return (SNMP_ERR_NOSUCHNAME);
}
```

- Inside those functions you typically add two switch operators - one on the SNMP operation to perform - SNMP_OP_GET, SNMP_OP_GETNEXT, SNMP_OP_SET, SNMP_OP_ROLLBACK, SNMP_OP_COMMIT and one on the leaf object whose value is requested / set

6) Test it and send a patch

- ports/net-mgmt/bsnmptools
- Simply doing a walk on the MIB subtree is not enough
- Each module is also documented - one needs to get his hands dirty with nroff(1) and mdoc to write a man page)
- FreeBSD developers usually prefer unified diffs but bsnmpd(1) modules are usually self contained and may be easier to mail a tarball
- Eventually the patch will be reviewed and committed to CURRENT

Available modules

- "ls -l /usr/src/usr.sbin/bsnmpd/modules/" - ops - not that many
- snmp_atm(3) - monitoring ATM interfaces
- snmp_bridge(3) - implements RFC 4188, RFC 4318 and more
- snmp_hostres(3) - Host resources - RFC 2790
- snmp_mibII(3) - one of the required modules - monitoring network interfaces, etc
- snmp_netgraph(3) - play with netgraph(4) via SNMP
- snmp_pf(3) - ops - that one is not documented, handy on machines using PF as a firewall
- also - bsnmp-regex - available in ports
- Smux - more information on http://wiki.freebsd.org/SnmpSmux

Ongoing work

- http://wiki.freebsd.org/BsnmpTODO
- Loadable transports for bsnmpd(1) - that is SNMP over Ethernet, ATM, SCTP , etc
- IEEE802.11 module
- EtherLike-MIB
- if_vlan(4) module
- SNMP access to pf ALTQ data
- Extend snmp_netgraph(3) module to allow creation and deletion of nodes and hooks via SNMP

Future cool stuff

- SNMPv3 support - a must - but requires a lot of work and proper design
- BEGEMOT-JAIL-MIB
- bsnmptrapd
- Sensors MIB Module
- lagg(4) module
- IPSEC module (RFC 4807)
- …

Demo - creating and configuring a filtering bridge with SNMP

```
#
# Bridge module
#
begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so"
```

- ✓ edit snmpd.conf to load the bridge module, start bsnmpd

```
#sudo /usr/sbin/bsnmpd -c /home/syrinx/snmpd.config
#bsnmpwalk -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
  begemotBridge

#man snmp_bridge
```

- ✓ to see what we can do with the module

Demo (2)

- Create bridge with name bridge1 and add a bge0 interface to it, also start RSTP on it

```
#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeBaseStatus[bridge1]=createAndGo"

#bsnmpwalk -s tryset@ ifTable | grep bge
ifDescr[4] = bge0

#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeBasePortStatus[bridge1, 4]=createAndWait"

#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeBaseSpanEnabled[bridge1, 4]=disabled"

#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeBasePortStatus[bridge1, 4]=active"
```

- Verify with ifconfig(8) output

Demo (3)

- What about dot1dBridge?

```
#bsnmpwalk -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
dot1dBridge
mib_2.17 = No Such Object
```

- dot1dBridge subtree is still supported but you either have to name your bridge interface - "bridge0" or explicitly change it

```
#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeDefaultBridgeIf=bridge1"
#bsnmpwalk -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
dot1dBridge
dot1dBaseBridgeAddress.0 = 5e:59:fd:ae:73:ae
dot1dBaseNumPorts.0 = 1
...
```

- Time to clean all the mess

```
#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeBasePortStatus[bridge1, 4]=destroy"
#bsnmpset -s tryset@ -i /usr/share/snmp/defs/bridge_tree.def
"begemotBridgeBaseStatus[bridge1]=destroy"
```

Thank you!

Questions?