parse `snmp.conf` configuration files allowing Perl scripts to make use of administratively defined configuration settings. The newer Perl modules are also somewhat more modular, and are grouped under the package name "Net-SNMP." This even includes a DBI-style wrapper module, providing access to SNMP management information via SQL queries. This comes with a "network shell" providing a number of useful additional features, such as aliasing, command definition, CSV and XML output etc.

One other addition to the agent, and something that has been missing for some time, is initial support for the `DISMAN-EVENT-MIB` from the Distributed Management working group. It has long been a source of confusion to people using the software that although the agent can be configured with various thresholds for "normal" behavior, it does not actually do anything when the activity being monitored strays outside these boundaries. Although the `DISMAN-EVENT-MIB` module is still very new and not all the features of it are supported, the agent can now take a much more active role in monitoring the system and can generate SNMP notifications when appropriate.

Software Organization

There have been other changes, including a reorganization of the header file structure, to avoid the irritating inconsistencies between coding within the main source tree, and using a fully-installed system. This also includes a number of wrapper header files, so application writers need only include two header files (or three if developing code for the agent).

There is a new tool (`net-snmp-config`) to help with identifying details of the configuration of an installed Net-SNMP suite. This shell script takes care of many previously annoying problems with compiling external applications and sub-agents. Makefiles for other packages can now just include a `net-snmp-config` invocation in order to get the appropriate list of libraries and header file paths to include in compilation rules. The `net-snmp-config` script will even compile a C-coded Net-SNMP agent handler module directly into an AgentX sub-agent in one invocation. Finally, it will perform other useful tasks like creating new SNMPv3 users – prompting for any necessary information.

And of course there have been innumerable bug-fixes and the like, as well as minor new features like long-form command line options. The code has also been generally cleaned up (indent has been run on the entire package to provide a vaguely consistent coding style) and more documentation has been written (much of it using the new adopted doxygen style of in-line-coding documentation). But this probably covers the main important changes and developments since the previous UCD-SNMP line releases.

Future Developments

And what about the future? Probably the main task on the horizon is a fuller review and re-design of the library – extending the modularization throughout the rest of the basic library APIs. This should aid those needing to extract individual portions of the library (e.g. to only include SNMPv3 support, or to omit MIB file handling when working in a resource-limited environment), as well as hopefully simplifying things for the general SNMP developer.

The `UCD-SNMP-MIB` is still supported, but experience has shown that at least some of the tables could benefit from a re-design. Any tables or groups that are rewritten will be moved into the new `netSnmp` enterprise branch of the MIB tree.

Then there are the moves afoot in the SMIng and EoS working groups looking at possible future developments for SNMP-based management, which are being actively followed by some of the Net-SNMP coders. The package is often a test-bed for new features of the SNMP framework, and we are hoping to be on the forefront of the EoS protocol implementations. If the timing is right, maybe the new Net-SNMP MIBs will be the some of first to make use of the new SMIv3 language.

And we have always had something of a bias toward Unix-based systems, and could desperately do with more input from those with experience of SNMP on Windows boxes. Although most elements of the package compile and run under Windows, with vaguely sensible results, the most active developers do not use Windows on a daily basis, and struggle to provide anything more than the most basic level of assistance.

It should be very clear that the new release is more than just a change of name, and the next few years show signs of being every bit as interesting as the last few have been. Why not come and join in?

Evaluating MIB II (RFC1213) Implementations

Henrik E. Holland, University of Twente
Remco van de Meent, University of Twente
Aiko Pras, University of Twente

Since the introduction of the Simple Network Management Protocol (SNMP) in the late nineteen eighties, SNMP and SNMP related network management has gone through substantial changes. The original version

of the protocol issued as RFC 1157, has already engendered two successors: SNMPv2c and SNMPv3. The basic architecture of the SNMP concept has however remained fairly intact as have its basic functions. One of these basic functions is to provide network management systems with the capabilities to monitor their networks.

It goes without saying that the accuracy of measured quantities and retrieved data values within a network is of great importance to the network operator. If a network operator cannot rely on the accuracy of the network statistics provided to him by means of some network management scheme, this scheme will quickly be rejected. In this respect the vast popularity of SNMP based network monitoring might reflect the (overall) strong faith which the networking community has bestowed in the accuracy of SNMP based network statistics. As with any network management scheme however, the use of SNMP involves some risk. This is the risk of perceiving reported statistics as accurate when they are in fact erroneous. In July 1991, three months after the MIB-II was issued as RFC 1213, a group of AT&T researchers at Bell Labs presented the results of a series of SNMP agent tests performed on nine different SNMP supporting routers [1]. The results of the experiments were somewhat disappointing as they showed that the risk of receiving erroneous data values was by no means neglectable.

A decade later, while celebrating the tenth anniversary of the MIB-II, we decided to reinvestigate the accuracy and performance of SNMP based network monitoring. Within this perspective, we had grown curious about the results of ten years of SNMP development and implementation. The research was carried out as part of the Dutch Internet Next Generation project and was produced under deliverable D2.16 [2]. During our investigation we evaluated the SNMP capabilities of three devices configured as Internet Protocol (IP) routers. Our research was dominated by three concerns:

1. The accuracy of statistics collected by the SNMP agent and reported to the manager.
2. The speed at which these statistics are updated.
3. The performance of the SNMP entity while the router is handling a heavy traffic stream.

We believe that these three concerns address the most basic issues in relation to SNMP based network monitoring, and will in this respect provide at least a general insight into the current status of MIB-II implementations. Note that we concentrated on counters and that we did not look at set operations, the generation of notifications, correct get-next processing etc.

The rest of this article is divided into three sections. The first section describes which part (i.e. which set of objects) of the MIB-II was tested, how we engineered the test set-up and the tests themselves. A subsequent section presents the test results and a bit of result analysis. The conclusions of the research are presented in the last section of this article.

Approach of the Evaluation

This section consists of three sub-sections, which as a whole describe the approach of the evaluation. The first sub-section describes the tested MIB area, the second presents the test set-up. The third section finally describes specifically the three types of tests conducted on each of the three routers.

The Test Objects

The first question requiring an answer was the selection of the MIB-II test objects. This question is an important one since its outcome determines the exact test configuration (e.g. the required hardware) and the way in which the tests are run. In total, we selected thirty-six objects from five MIB groups:

```
system:                 ip:
  sysDescr                ipInReceives
  sysObjectID             ipInHdrErrors
  sysUpTime               ipInAddrErrors
  sysContact              ipForwDatagrams
  sysName                 ipInUnknownProtos
  sysLocation             ipInDiscards
  sysServices             ipInReceives
                          ipInDelivers
interfaces:               ipOutRequests
  ifInOctets              ipOutDiscards
  ifInUcastPkts           ipOutNoRoutes
  ifInNUcastPkts          ipRoutingDiscards
  ifInDiscards
  ifInErrors            udp:
  ifInUnknownProtos       udpInDatagrams
  ifOutOctets             udpOutDatagrams
  ifOutUcastPkts
  ifOutNUcastPkts       snmp:
  ifOutDiscards           snmpInPkts
  ifOutErrors             snmpOutPkts
                          snmpInGetRequests
                          snmpOutGetResponses
```

The `interfaces` and `ip` group objects shown in the table above were selected because they belong to the set of MIB-II objects frequently used for SNMP based network monitoring. During the object selection, openly available vendor recommendations (such as [3]) considering the

use of specific MIB-II objects were taken into account. Specifically included into the evaluation were objects responsible for delivering error related statistics such as cyclic redundancy check (CRC) error counters and IP header error counters. Finally, to achieve a complete overview of the MIB-II implementations, we included a few objects from the `udp` and `snmp` groups. All but the objects from the `system` group are of the `Counter` data type as defined in RFC 1155.

The Test Set-Up

To address our concerns accordingly, we devised a test configuration which allowed for an isolated evaluation of our devices. This excluded the possibility of network traffic interfering with our tests. The test set-up is shown in Figure 1.

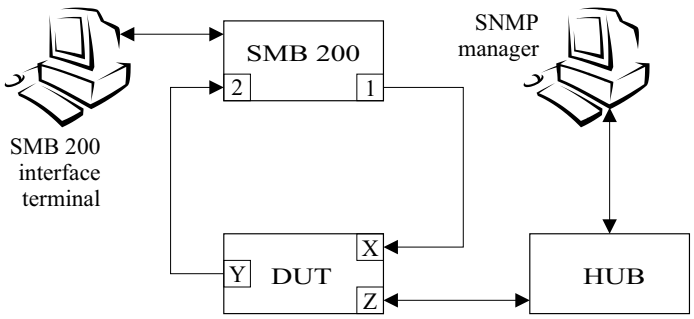

Figure 1: Test Set-Up.

All devices in the test-setup were connected with standard cat-5 cables. The devices unter test (DUTs) were the following:

- A: Cisco Systems AGS+, software: GS3-K Version 9.1(11), ©1986-1994.
- B: Cisco Systems C2600, software: IOS C2600-I-M Version 12.0(3)T3, ©1986-1999.
- C: Cabletron Systems ssr2000, software: Riverstone Networks Version 6.2.0.1, ©2000.

For the measurements, the following additional devices have been used:

- SMB200: Netcom Systems Smartbits 200 with two ML-7710 Ethernet smartcards.
- SNMP manager: Adventnet SNMP Utilities 3.0 on Windows NT.

From now on we shall refer to the routers as A, B and C respectively. Note that routers B and C are 100 Mbps devices while router A is a 10 Mbps device. This older 10 Mbps router was chosen for the evaluation to see if the SNMP capabilities of newer devices have improved.

Figure 1 shows that the device under test is connected to the Smartbits 200 over ports `x` and `y` and to the Adventnet SNMP manager via port `z`. The general idea was to use the Smartbits for generating and transmitting IP traffic, and to poll for MIB objects with the Adventnet manager. The Smartbits 200 is an industry standard transceiver with advanced network data generation and measurement capabilities [4]. Amongst other things, it allows the user to set up layer two and three traffic streams and it provides the user with frame editing capabilities. Furthermore, the Smartbits is capable of accurately counting and analyzing received bytes and frames. Tags can be added to IP packets before transmission to ensure that only packets transmitted by the Smartbits itself are counted at the receiving end.

As stated above, our test set-up was to enable an isolated evaluation of the test devices, so that no network or arbitrary background traffic could pollute our test results by unintentionally triggering the MIB-II counters. However, during the configuration of the test set-up, we found that router B was emitting some kind of spurious background traffic. Regrettably, we were unable to halt the transmission of these frames. Although the frequency of the frames was very low, it did cause some of the MIB object counters to increment. During our analysis, we canceled out the error introduced by the disturbing traffic by measuring its frequency and the size of the frames, and correcting the collected MIB object values accordingly.

Three Types of Tests

Within the scope of the evaluation, we have distinguished three kinds of test sets and henceforth three types of outcome spaces. The test sets and outcome spaces each correspond to one of the earlier defined concerns.

Accuracy of the Statistics

The set of tests devised to determine the accuracy of the statistics involved routing IP datagrams through the DUT and polling the MIB objects before and after the respective transmission.

The tests were executed by transmitting large packet bursts of different frame sizes at various transmission speeds. The exact content of the test streams was configured in accordance to the object(s) tested during that particular test run. In general, there were two kinds of test frames: non-erroneous Ethernet frames containing normal non-erroneous IP packets, and frames containing some kind of error where a distinction is made between an Ethernet frame error and an IP header or IP address error. The erroneous frames were con-

structed manually using the Smartbits' frame editor. In this respect, the Smartbits did limit our capabilities slightly. We were not able to put together a traffic burst consisting of an arbitrary amount of different Ethernet frames or IP packets. Instead the Smartbits allows one customized frame or packet to be transmitted along with the stream every `x` normal data frames, were `x` is a positive natural number.

As noted in the previous paragraph, the burst sizes of the test transmissions were very large. Depending on the transmission rate, the burst sizes varied between 5,000,000 and 10,000,000 packets. As does the correction for background traffic, transmitting such large amounts of packets will contribute to the accuracy of the test result. The unwanted incrementation of counter values due to the polling of the DUT's SNMP entity before and after the test becomes less of a problem, since the amount of SNMP packets belonging to the polls is relatively small.

The size of the test frames was constant during each individual test run, but varied per test run. Test runs of two different frame sizes were made on router A, and of three different frame sizes on router B and C. The frame sizes used were 64, 512, 1024 and 1518 bytes.

The rates of the test transmissions expressed as a percentage of the utilization rate, varied between 40% and 100% for routers A and C, and between 20% and 100% for router B. The utilization rate, here in respect to Ethernet, refers to the maximum theoretical transmission rate, which depends on the size of the transmitted frames.

Each device was tested at least at three transmission rates, where a distinction is made between two situations. In the first situation, the router is not dropping any of the offered packets while in the second situation it is. The last situation can occur when the device has to handle a large amount of traffic. Our intention was to test the devices in both situations. To do this precisely, we had to know at what rate the routers started dropping packets. The rate pertaining to this qualification is called throughput rate. According to RFC 1242, throughput is defined as follows:

> *Throughput: The maximum rate at which none of the offered frames are dropped by the device.*

Before commencing with the actual evaluation, the throughput rate of the devices was determined using the Smartbits' benchmarking suite. These tests were conducted conform the benchmarking methodology as described in RFC 1944. Table 1 shows the results of these tests.

The results of the throughput tests not only provided a qualification necessary for the tests themselves, but they also revealed the "class" of the devices. B is obviously of lesser cachet than A and C, as far as throughput is concerned. These results are however not so awkward considering the difference in price between the devices.

router	throughput rate (pps)	theoretical max. throughput rate for Ethernet (pps)
A	10020	14880
B	25227	148810
C	147932	148810

Table 1: IP Throughput Rates (64 bytes per frame).

Update of Counter Values

The second series of tests was conducted to verify the speed with which new MIB object values are made available. These tests were only run on objects of the `Counter` type. We used the same types of data streams as during the first test series, but now the MIB values were polled *during* the transmissions and not only before and after.

The objects were polled at a frequency of one poll per second. The difference between two successive values was plotted in a graph in real time during the test transmission. A continuous counter update within the DUT should therefore reveal a straight horizontal line representing the transmission rate in bytes or packets per second, depending on the polled object.

Of course there will be some fluctuation in the exact moment of the poll. Under normal circumstances however this fluctuation will never be large relative to the poll interval of one second. This test was not performed to measure very exactly certain values at certain moments and thus a small variation in the results is acceptable. Furthermore, by looking at the difference between successive values and not at the absolute values some of the (constant) delay is canceled out.

These tests were performed apart from the first test series described above, since we did not want the continuous polling of the MIB objects to interfere with the packet counts.

Performance Under Heavy Load

During the third series of tests, the response of the DUT's SNMP entity was tested while the DUT was dropping packets. The test verified if object values could still be delivered while the DUT was under strain and if so how many polls could successfully be made per second. In effect this means that the rate of the test stream was configured to be higher than the DUT's throughput rate. The SNMP entity was polled at various intervals

during the tests. This poll interval was constant during a test run but it varied between the test runs. The IP test stream offered to the device consisted of 512 byte Ethernet frames at a transmission rate of up to 100% of the utilization rate.

Results

As described in the previous section, the evaluation consisted of three types of tests resulting in three sets of outcome spaces. Each of these outcome spaces will be treated separately in the following sections.

Accuracy of Counter Values

The accuracy tests in themselves were concluded successfully as we were able to obtain useful and significant test results. With a few exceptions, the accuracy of the octet and packet counts was high for all of the evaluated routers. All of the SNMP entities reported counter values which stayed within a margin of at least 1 % of the actual amount of offered octets or (respectively) packets.

The mentioned exceptions are related to Router C. The tests showed that the router's `ipInReceives` and `ipForwDatagrams` counters are troubled by a structural miscount of 2 packets for every counted packet. This means that for every incoming IP packet, three are counted.

In addition to the "triple count" error encountered at router C's SNMP entity, a number of specific peculiarities were observed, which concern routers A and B as well. The next sections will describe these peculiarities.

Packet Loss Count

During our tests, routers A and B dropped packets when the test stream was set to a higher rate than their respective throughput rate. The dropped packets, however, could not be accounted for at any of the counters belonging to the `interfaces` or `ip` groups.

There are two objects within the `interfaces` group and two within the `ip` group, which are related to the discard of packets not due to erroneous input but (amongst others) to a lack of buffer space:

- `ifInDiscards`
- `ifOutDiscards`
- `ipInDiscards`
- `ipOutDiscards`

When an IP packet is discarded, it should be counted at a counter pertaining to one of the above listed objects depending on where it is discarded. If the packet is discarded at IP level, then it must be counted at `ipInReceives` or `ipForwDatagrams` as well, depending on where it is discarded. To illustrate the above, Figure 2 shows a part of the `ip` group case diagram as depicted in [5]. Only the objects relevant for this example have been included, the dotted lines serve to indicate where objects have been left out.

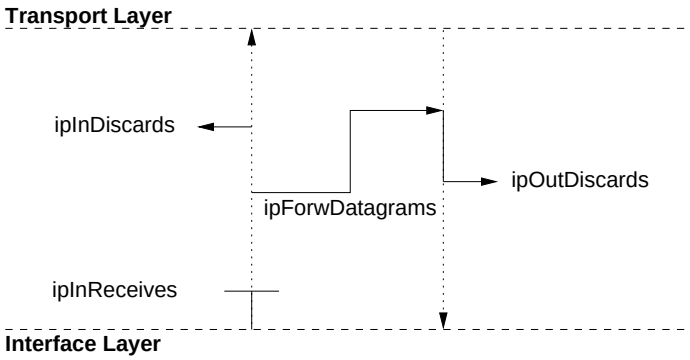

Figure 2: Excerpt from the IP Group Case Diagram.

During our tests, none of the discarded packets were counted at `ipInReceives`. This implies that the packets were not discarded at the IP level, apparently we must focus our attention on the lower interface layer.

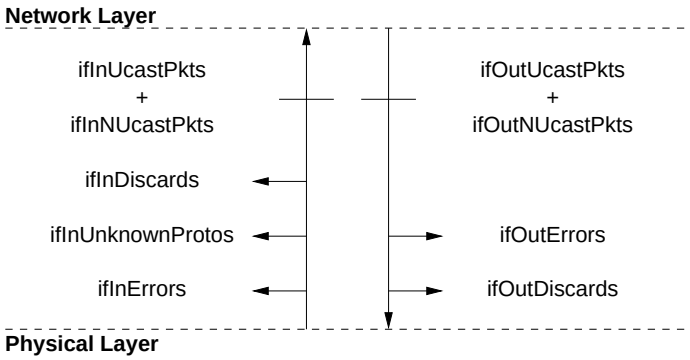

Figure 3: Interfaces Group Case Diagram.

Figure 3 taken from [5] shows the `interfaces` group case diagram. Remarkably enough, `ifInOctets` is not depicted in this case diagram. Curiously we took a look at [6] but found that the `interfaces` group case diagram presented in this book, also lacks the `ifInOctets` object. This seems at least awkward. RFC 1213 defines the `ifInOctets` object as follows:

> `ifInOctets`*: The total number of octets received on the interface, including framing characters.*

The definition implies that `ifInOctets` should count *all* of the octets received on the interface. In the case diagram, this would place the object just above the

physical layer, and under the `ifInDiscards` object. This means that just as `ipInReceives` should count packets before they are discarded at the IP layer, `ifInOctets` should count all of the octets at the interface layer before being discarded. During our tests, however, this was not the case.

Apparently, the frames are dropped before reaching any of the counter mechanisms pertaining to the mentioned objects. Although this might be understandable, a network operator will want to see the amount of packets dropped by the devices that he manages. Besides this, the presence of the discard objects, defined as they are, bestows the network operator with the belief that he is able to monitor the amount of dropped packets, which is apparently not the case for routers A and B.

Router C did not drop a significant amount of packets during any of the test runs, since its throughput rate is almost equal to the utilization rate at all of the possible frame sizes. Henceforth, we could not study the administration of dropped frames for this device.

CRC Count

The evaluation revealed that routers A and B count incoming frame octets excluding the four byte Ethernet cyclic redundancy check. Router C however, does include the four byte CRC into its `ifInOctets` count.

The definition of `ifInOctets` suggests that the octet count should include the four byte CRC, since the count must include all of the octets received on the interface, including the framing characters. The implementation of this object in routers A and B therefore seems wrong.

Count of Packets With IP Header and Address Errors

During the evaluation we simulated four kinds of IP address errors. Two of these types (invalid and reserved Class E) should have lead to an increment of `ipInAddrErrors`, while the other two (not routable and multicast) were meant to test the `ipOutNoRoutes` object. We also tried various kinds of IP header errors to test the `ipInHdrErrors` object.

The IP header errors were handled and counted accurately at all of the right objects by all of the devices. Handling and counting IP packets with an invalid IP destination address proved to be somewhat more of a problem.

All of the tested routers accurately count the offered IP address error test packets at `ipInReceives`, which is a correct procedure. Just as well, all of the test packets are found to be erroneous by all of the routers and are henceforth discarded. What differs between the routers' behavior is the use of the `ipInAddrErrors` and `ipOutNoRoutes` objects. Router A discards the test packets, nonetheless neither `ipInAddrErrors` nor `ipOutNoRoutes` is incremented as it should. One type of address error was counted at `ipInUnknownProtos`, which is incorrect as well. Device B does a "better" job: it correctly counts two out of four test packets at `ipOutNoRoutes`. Router C does not count any of the test packets at respectively `ipInAddrErrors` or `ipOutNoRoutes`.

Count of CRC Erroneous Frames

Routers A and B count frames with an erroneous CRC value solely at `ifInErrors`. However, the definition of `ifInOctets` suggests that the erroneous frames should be counted at the `ifInOctets` counter as well. Routers A and B thus function incorrectly in this respect.

Router C also exhibits faulty behavior with respect to the count of the cyclic redundancy check characters. Router C counts the CRC erroneous frames at `ifInOctets`, `ifInErrors` and `ipInReceives` as well. The erroneous packets should however be discarded after being counted at `ifInErrors`, and should thus never reach the IP layer and its pertaining counters.

Wrap Around Errors

The `ifInOctets` and `ifOutOctets` object counters of router C are affected by wrap around errors. Our tests revealed that these errors occur more than occasionally. During six out of eight test runs, one or more of the mentioned interface objects proved to contain an erroneous counter value.

Our test results showed that the counters seem to "stick" for a while before proceeding with their count. The value at which the objects stick is the maximum value for a 32-bit counter. We have not determined exactly how long the counters stay at this value. It is possible that the wrap around errors are due to a counter type conflict. 32-bit counters are polled, while 64-bit counters are most probably in use on this device.

Counter Updates

Our tests showed that all of the routers' SNMP entities could provide actual statistics at a frequency of one Hertz. There was one exception which concerned router A and the `ifInErrors` object. The data value pertaining to this object changed 24 times every four minutes, which corresponds to one change per ten second. No other rarities were observed with any of the routers during any of the test runs. We may therefore conclude that all three routers could easily maintain their counters in a timely fashion.

test run	IP rate (pps) (%)	SNMP rate (pps)	SNMP loss (%)
1	5,000 (21)	10.0	0
2	10,000 (43)	2.5	72 ± 3
3	10,000 (43)	10.0	93 ± 3
4	20,000 (85)	10.0	94 ± 3

Table 2: SNMP Packet Loss under Heavy Load.

Object Retrieval Under Heavy Load

As described in a previous section, routers A and C are capable of realizing high throughput performance, whereas router B is much more limited. It seems that this difference reflects in the performance of the devices' SNMP entities when the device itself is under heavy load. Routers A and C do not have any problems with reporting requested object values while handling a large amount of traffic. However, router B shows a great performance degradation of its SNMP entity. Even when the rate of the IP test stream is below the device's throughput rate, the device is unable to report all of the requested object values.

Table 2 shows a list of test runs performed on router B during this series of tests. In column two, it shows the transmission rate of the IP test stream in packets per second (pps) and as a percentage of the utilization rate. The third column shows the amount of SNMP requests per second in packets per second (every IP packet contains one request). The last column contains the percentage of SNMP requests not replied to for each run. During runs one through three, the rate of the test stream was below the throughput rate of the router. The figures in the last column show that the device was unable to respond to all of the SNMP requests during runs two, three and four.

What we wanted to see is where exactly these SNMP requests are lost, i.e. where they appear in the packet counts and where they become absent. To illustrate this, Figure 4 shows a flow diagram containing relevant sections from the UDP and SNMP case diagrams.

The diagram is not meant to provide a complete overview of the respective case diagrams including all pertaining objects, but to illustrate where the SNMP packets should be counted and what the relation is between the counters. Printed beside the objects in Figure 4 are the results from test run two. The figure shows that somewhere between `udpInDatagrams` and `snmpInPkts` the requests "disappear" from the statistics, this area is encircled.

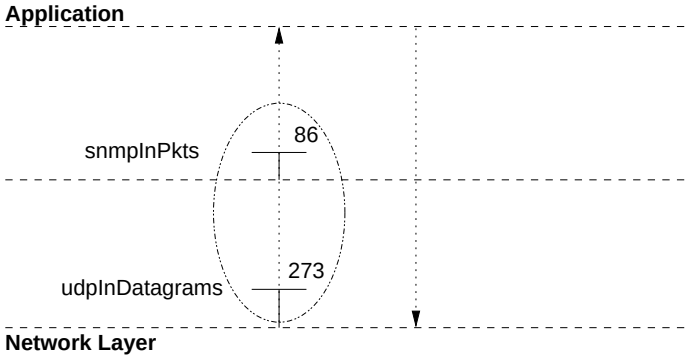

Figure 4: Example of the SNMP Packet Flow.

Conclusion

The evaluation described in the previous sections proved to be an interesting one. Our tests were dominated by three concerns: accuracy of the counters, speed of counter updates, and the performance of the SNMP entity under heavy load. These three concerns provided a satisfactory basis for the tests, and as a whole formed a suitable framework.

Our tests showed that the overall accuracy of the evaluated SNMP entities was good. Where packets were counted, this was done very precisely and consistently. There were however a number of errors concerning some basic objects. Most striking is the two packet miscount noticed at two of router C's `ip` group objects. This device also exhibits wrap around errors at two of its `interface` group objects. Interesting as well is that router C does count Ethernet CRC bytes, while routers A and B (incorrectly) do not. More inconsistency between the routers was observed regarding `ifInOctets` in case of CRC errors. A and B discard frames with CRC errors without counting them at this object while the definition suggests that this is wrong. All of the devices appeared to have problems with handling IP packets with an IP address error. Only device B administrates two out of four address error types correctly.

With the exception of one object at router A, all of the tested routers were able to update all counter values at least once per second.

Router B did not exactly pass the performance test of its SNMP entity with flying colors. Even when the router is not subjected to a high rate IP test stream, it loses the ability to respond to all of the received SNMP requests. The other devices show no problems of this kind. They maintain their ability to respond to a large amount of SNMP requests per second, under all conditions.

If we compare our results to those of the Bell Labs tests from 1991, an overall improvement in accuracy can be observed. Nevertheless, a number of serious

errors were found. At the basis of some of these errors could be a simple inconsistency in viewpoints regarding exactly *how* some of the objects must be implemented, e.g. the CRC count discrepancy between the routers, and the miscount of CRC erroneous frames on routers A and B. The MIB-II RFC, however, is clear about these particular issues, and a sound implementation according to the standard seams feasible. What will help vendors with a correct interpretation of the RFCs, is the inclusion of case diagrams within these RFCs. This would unambiguously define how and where each object should be implemented.

Our overall impression is that the basic objects, regarding incoming and outgoing octets and IP packets, are implemented quite well, although this is a cautious and reserved statement, keeping the errors found at router C in mind. The implementation of more specific error objects, such as `ipInAddrErrors`, is done rather poorly on all of the devices. Based on the results, it seems fair to say that operators should always verify the statistics provided by a device before putting them into use.

References

[1] Mier, E., Bell Labs Test Routers' SNMP Agents, Network World 8(26), July 1991

[2] Holland, H.E., Evaluating MIB-2 Implementations, Deliverable D 2.16 of the Internet Next Generation Project

[3] Cisco Systems, Cisco Network Monitoring and Event Correlation Guidelines

[4] Spirent Communications, Smartbits 200, December 2001

[5] Rose, M.T., The Simple Book, An Introduction To Internet Management, 2nd Edition, Prentice-Hall, 1994

[6] Stallings, W., SNMP, SNMPv2, SNMPv3 and RMON 3rd Edition, Addison Wesley, 1999

The Story Behind the SNMP Vulnerabilities

Tiina Havana, OUSPG
Ari Takanen, OUSPG

In the PROTOS project, the Oulu University Secure Programming Group (OUSPG) and VTT Technical Research Center of Finland have developed and used software testing methods in a new application area: in the discovery of security problems in software. Since 1996, the researchers in Oulu have effectively been studying methods for pro-actively discovering security related bugs in software. During the past years, several security and reliability test-suites have been developed for various protocols by PROTOS researchers.

In spring 2002, the PROTOS project released a new test suite that received a lot of publicity. This time, the research group focused on SNMP. It was chosen to be the next test material after several flaws in various LDAP implementations were found using the PROTOS testing methods. Like LDAP, SNMP utilizes complex ASN.1/BER structures. Many of the discovered SNMP and LDAP flaws were caused by errors in decoder functions. Exceptionally formatted BER constructs, such as length fields claiming 2GB data to come, caused havoc amongst BER implementations. Different kind of errors were observed when correctly encoded SNMP packets containing exceptional data passed the decoding layer and proceeded to the application layer of a SNMP implementation, long community names triggering buffer overflows being a perfect example.

SNMP implementations are a good choice for trying new testing approaches in practice. SNMP is an old and mature protocol with numerous vendors providing solutions for it and even more numerous parties providing critical services over it. SNMP is also a complex protocol using an error prone ASN.1 encoding and containing various error prone data types over an unreliable network environment (UDP). Unlike most of the other testing groups who select a vendor and test the vendors products, OUSPG looks for interesting protocols for testing – and SNMP was one. Eventually, with the help of the testing tool developed in the PROTOS-project during the past couple of years it was possible to prove that SNMP implementations were vulnerable and could be exploited if wanted.

There be Bugs

In the SNMP-case about ten software products were tested by OUSPG researchers, but dozens of others were tested by the software developers themselves. The sample size of the products that OUSPG decided to test was intentionally limited. The programs that they chose for testing were selected according to their knowledge of programs that use SNMP and the availability of those programs. Products were not tested if no evaluation copy of the product was available, the evaluation copy had a restrictive license prohibiting evaluation or OUSPG simply was not aware of the product. OUSPG encouraged software developers and integrators to test their products by themselves.