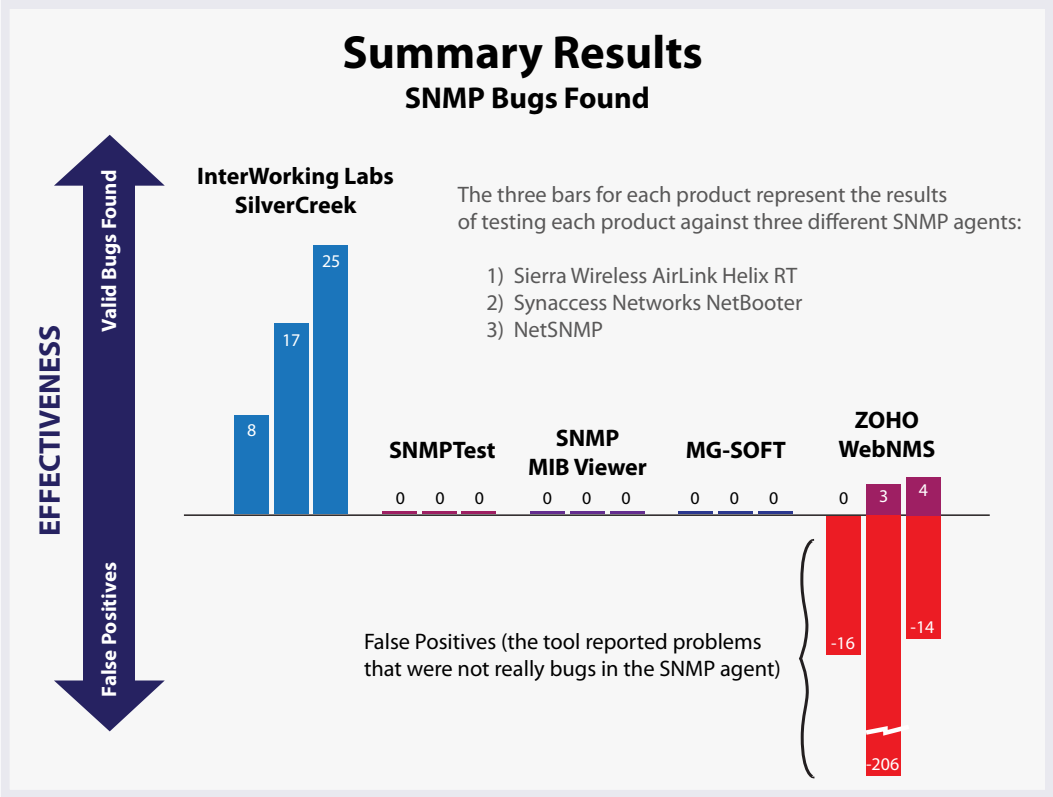
Comparison of SNMP Agent Test Tools

Executive Summary

The test engineer's job is to ensure bugs are found before products ship, thus saving the company money and helping to build the company's reputation for shipping quality products. Research has shown that fixing software problems after products have shipped can be up to 32-times more costly than finding and fixing bugs during a product's testing lifecycle[1]. For this reason, forward thinking organizations understand the importance of system test and the payback that results from doing comprehensive testing before products ship.

Simple Network Management Protocol (SNMP) is a complicated body of functionality that enables products to be remotely monitored and managed. There are a number of approaches to testing this capability and I set out in this report to experiment with the various alternatives to see how they might differ in delivering results.



SilverCreek consistently found more bugs without generating false positives

Surprisingly, I found a whole suite of common testing approaches that found absolutely no bugs at all (thus completely failing to perform the job) or worse yet, reported false positives that turned out not to be bugs at all (hence requiring many hours of time to determine and prove that the reported errors are not real bugs). Testing with these tools would leave an organization with a false sense of accomplishment and security until customers start calling in problems.

Of all the products I tried only InterWorking Labs' SilverCreek product proved effective at finding legitimate bugs.

1 NIST analysis of Baziuk 1995 study. See http://www.nist.gov/director/planning/upload/report02-3.pdf

Introduction

If you're in the business of developing or testing a product that supports the Simple Network Management Protocol (SNMP), I'm sure you've become aware of just how big a job it can be to test and verify new implementations as well as perform ongoing regression testing.

InterWorking Labs (IWL), the industry pioneer in SNMP test tools, recently commissioned me to play the part of an SNMP test engineer. It seems that a broad range of SNMP products are starting to be marketed as SNMP test and verification tools and IWL asked me to survey the market and test a complete range of available offerings.

This report provides an overview of my experiences which varied widely and yielded some surprising results.

Approach

I started my search much like you might – with a google search for "SNMP test tools". I found a range of offerings and I weaned my shortlist down to those tools claiming to be useful for test and debug.

I settled on five tools I thought best matched my criteria for this research. These included:

- InterWorking Labs' SilverCreek
- Paessler SNMP Test
- SNMP MIB Viewer
- MG-SOFT's Professional
- ZOHO's WebNMS

Product Feature Comparison

		InterWorking Labs SilverCreek	Paessler SNMPTest	SNMP MIB Viewer	MG-SOFT Professional	ZOHO WebNMS
MIB Walk		✓	Minimal	✓	✓	✓
SNMP Reads (GET, NEXT, BULK)		✓	✓	✓	✓	✓
SNMP Writes (SETs)		✓		✓	✓	✓
SNMPv1 & v2c & v3		✓	✓	✓	✓	✓
Importing Private MIBs		✓		✓	✓	✓
Boundary Testing		✓				Minimal
Compliance Testing		✓				
Vulnerability Testing		✓				
Stress Testing		✓				Minimal
Batch Operation for Regression Testing		✓				✓
MIB Compliance Testing	MIB II	✓				
	Interface	✓				
	RMON I & II	✓				
	IPv6	✓				
	SNMPv3	✓				
	DOCSIS	✓			✓	
Cost		$$$$	Free	$	$$	$$$

Report Contents

To get started I selected a wireless router[2] to use as a common SNMP agent and I ran each of the SNMP testing tools against this agent to get a baseline reading on the differences between the tools. Read on to learn about my detailed experience using each tool to test this agent.

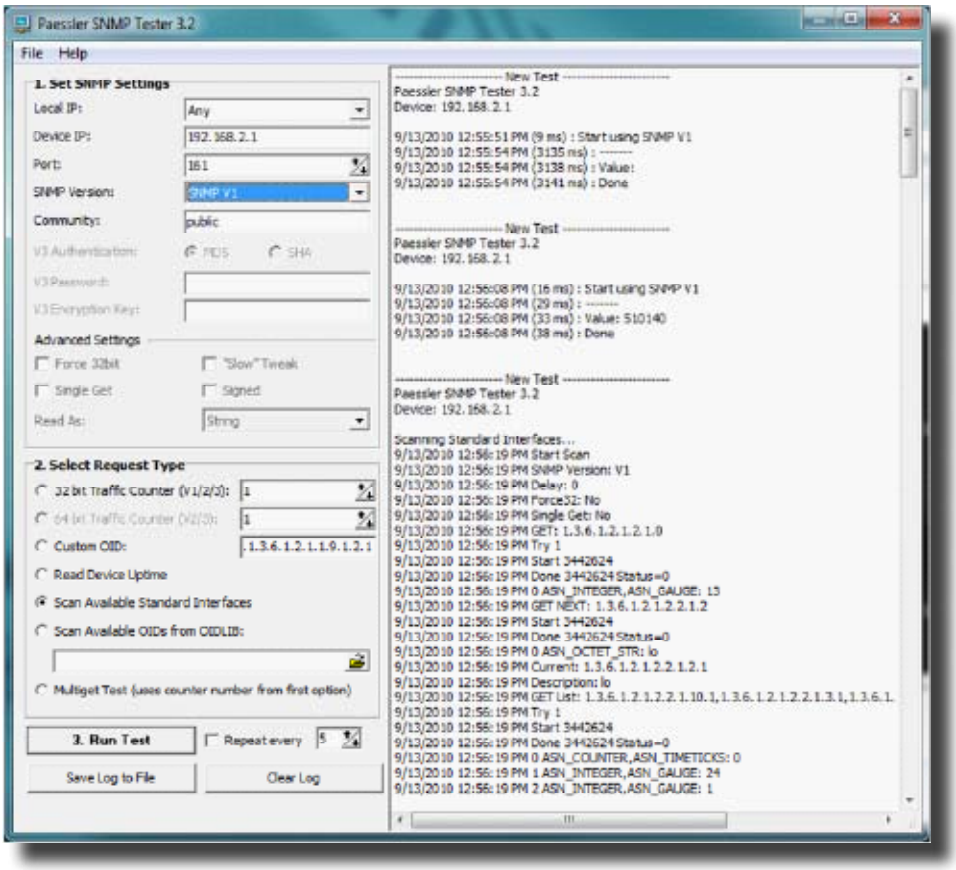
Detailed Experiences with Each of the SNMP Test Tools

Paessler SNMP Tester

Paessler's SNMP Tester (v3.2) seemed like a promising tool to start with. First of all, by name it is an 'SNMP Tester'. Secondly, the price seemed intriguing. The tool is freeware and is available as a free download.

Testing Results

I loaded Paessler SNMP Tester and found that it would only support individual retrieval of discrete SNMP object identifiers (OIDs), and that it requires the manual entry of the complete OID as well as the OID type. Due to the need to manually enter each OID, SNMP Test would be virtually unusable for thorough testing of a complete SNMP agent.



The program also supports some very basic scans for the device uptime and for scanning the standard SNMP interfaces. Unfortunately the tool found none of the agent bugs that were uncovered by subsequent testing.

SNMP MIB Viewer

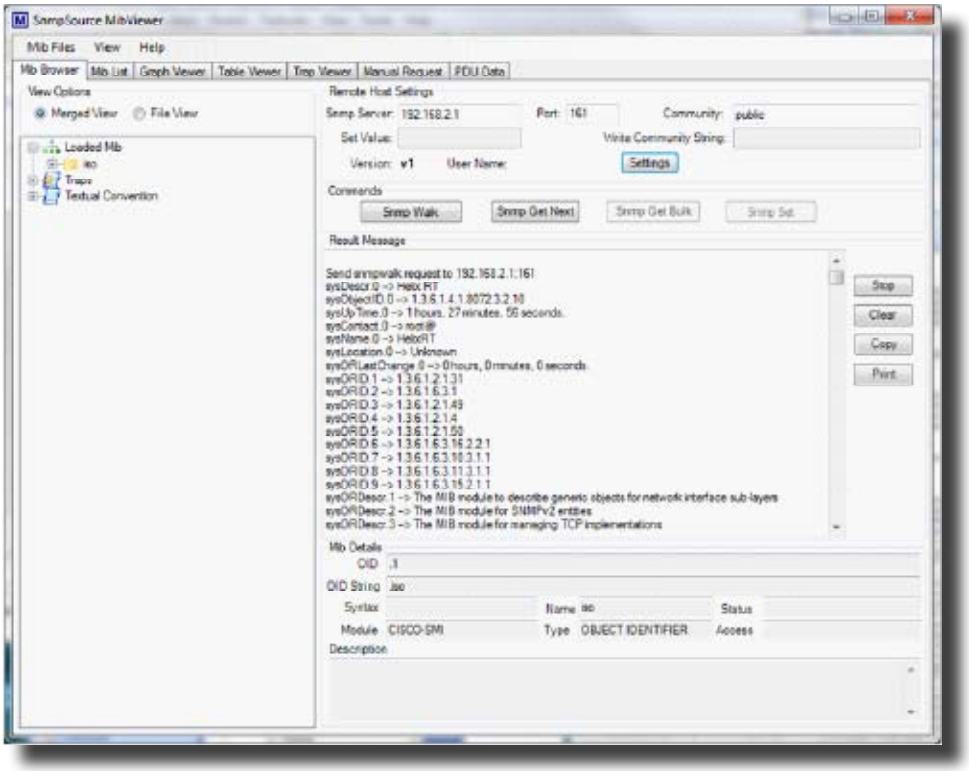
Next I tried SNMP MIB Viewer (v2.0.2), a shareware package available for $249.

Testing Results

I was able to load SNMP MIB Viewer on a windows 7 system without issue. There were virtually no instructions, and the application required the manual loading of each MIB one at a time. Each time a new MIB was added the IP address of the agent was reset.

2 I used the Sierra Wireless AirLink Helix RT with software version 4.1.0.010 for the body of this report, and two additional agents (NetBooter and NetSNMP) later in the testing process to verify the findings in additional test environments.

I was able to complete a full SNMP walk of the device under test. The walkthrough ran without issues except for reporting when individual OIDs were not in one of the loaded MIBs. No agent bugs were found and all indications were that the device under test was operating without error. If I had relied solely on a tool like this to perform my testing it is clear bugs would have gone undiscovered as confirmed by some of my later testing (read on).



MG-SOFT MIB Browser Professional Edition

MG-SOFT's MIB Browser Professional (version 12e 2010) comes in a broad and slightly confusing array of possible configurations. The version I tested supported SNMPv3 and DOCSIS, and is available for $999.

MG-SOFT Testing Notes

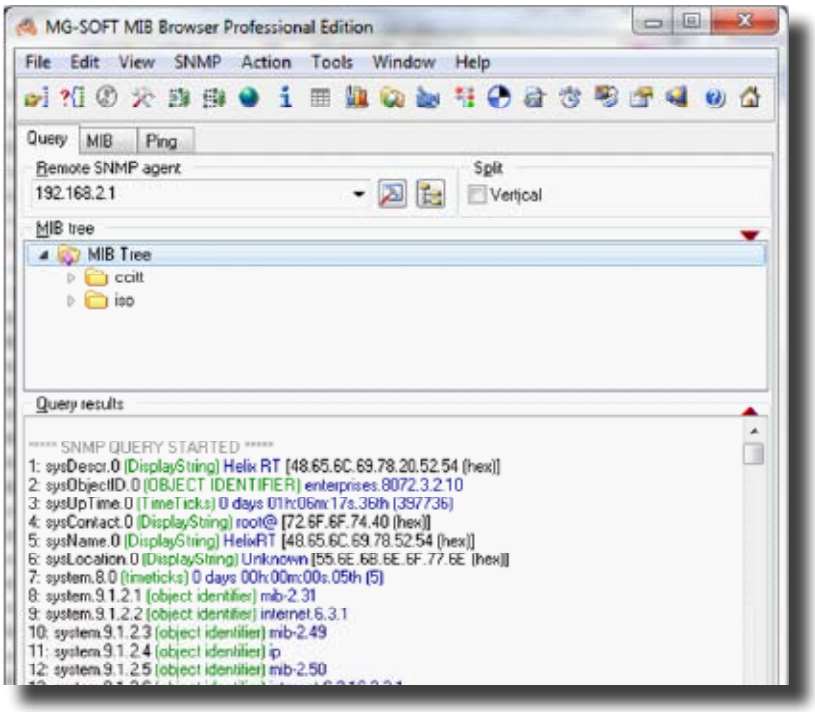
MG-SOFT's MIB Browser seems optimized to manage a number of SNMP agents on a network. Out of the box it starts automatically polling an arbitrary IP address. In order to get the tool to start running against my test device I need to configure it to poll for SNMP agents giving the tool a range of IP addresses from the subnet on which I had configured my test network. While this functionality might be welcomed if I were running a real network, I found this a cumbersome approach to doing system test in a lab environment.

Once I got the agent IP address configured, MG-SOFT automatically walked the MIB and showed me all information it recognized from the standard MIBs including MIB II and the Interfaces MIB (the tool calls this the 'standard info' window). MG-SOFT then automatically started polling the device every 60 seconds looking for updates. I could select a menu option to have MG-SOFT query the entire MIB Tree, and the tool did a credible job of retrieving and displaying an exhaustive list of both known and unknown MIB variables (unknowns are those OIDs for which a MIB had not been loaded).

Unfortunately MG-SOFT failed to find any of the agent bugs that I uncovered in later testing. While this tool seems like a viable low cost alternative for doing basic level SNMP monitoring of a network, this is clearly not a tool designed to test the limitations or completeness of an SNMP agent implementation.

ZOHO WebNMS Agent Tester

After failing to find any bugs at all with any of the previous tools I was looking forward to giving ZOHO's WebNMS a go (version 4). This tool, which is available for $1,194, described itself as "designed to test the SNMP Agents and the MIBs implemented in an agent (with) powerful built-in test cases (and) a complete test suite customized for the agent within minutes."



WebNMS Testing Notes

I installed WebNMS on a Windows 7 64-bit system. I suspect it has never been tested by the manufacturer under these conditions because the tool refused to install in the default folder 'Program Files(x86)' complaining about 'illegal characters' and aborted the installation. After some effort I figured out I could manually install it in the 'Program Files' directory to circumvent this issue.

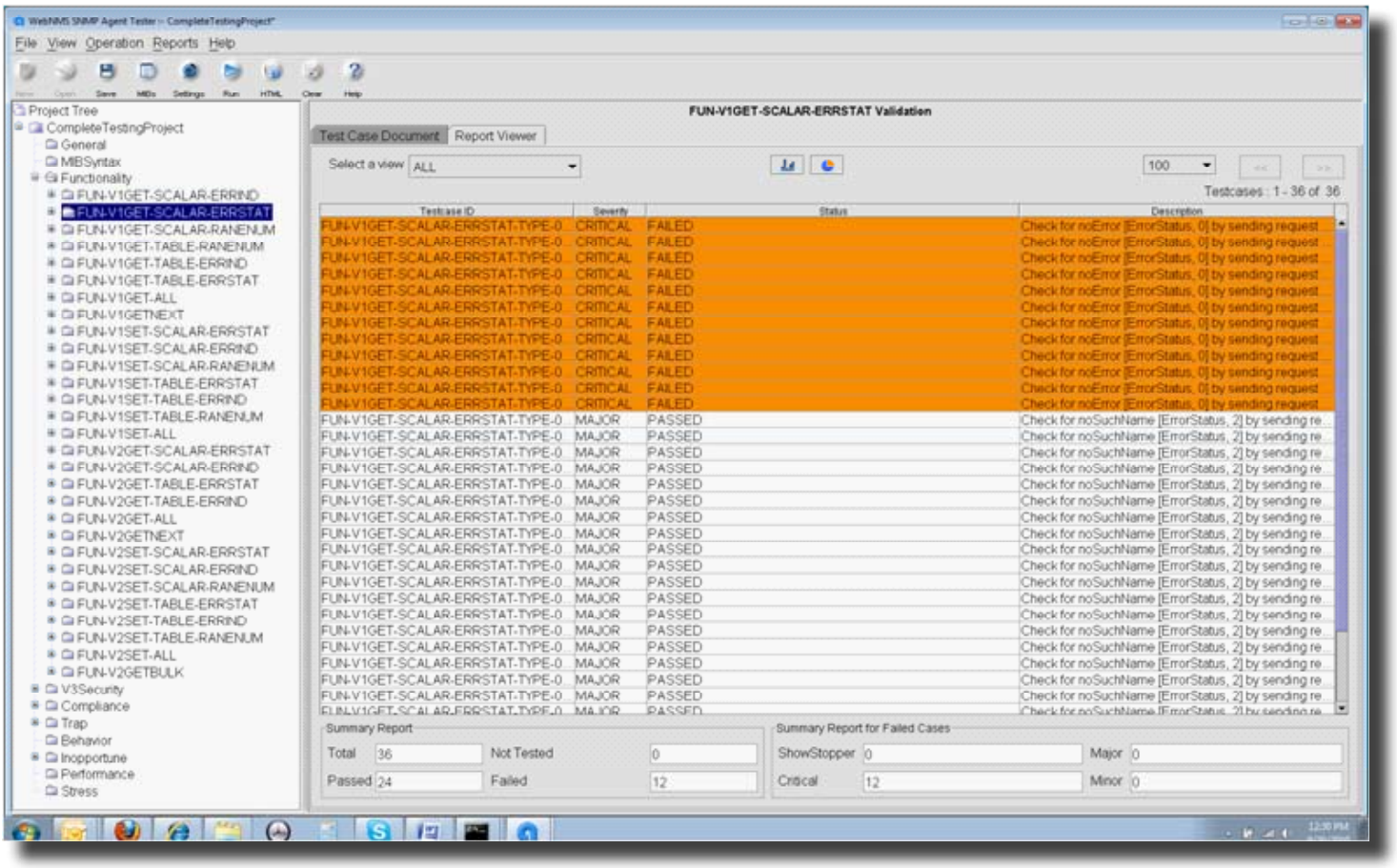
After this bumpy start I got the tool up and running and I entered the agent's IP address wherein the tool discovered the agent as v2c compatible. This was unfortunate since I don't think this agent supports v2c. I manually overrode the agent configuration to limit the testing to SNMPv1 read-only operations so that I could continue without generating extraneous errors.

WebNMS immediately identified lots of errors. I took this as a promising start, but after many painful hours digging through each of the failed tests I came to believe that every single issue identified by the tool was in fact a false positive. None of these were real bugs in the agent.

WebNMS's main interface reported a number of errors that after much investigation I've come to believe are all interrelated and are all false positives – meaning they are not real bugs.

As an example, the very first Functionality test (FUN-V1GET-SCALAR-ERRSTAT-TYPE-01-001) reported a CRITICAL error. The tool reported the test details as:

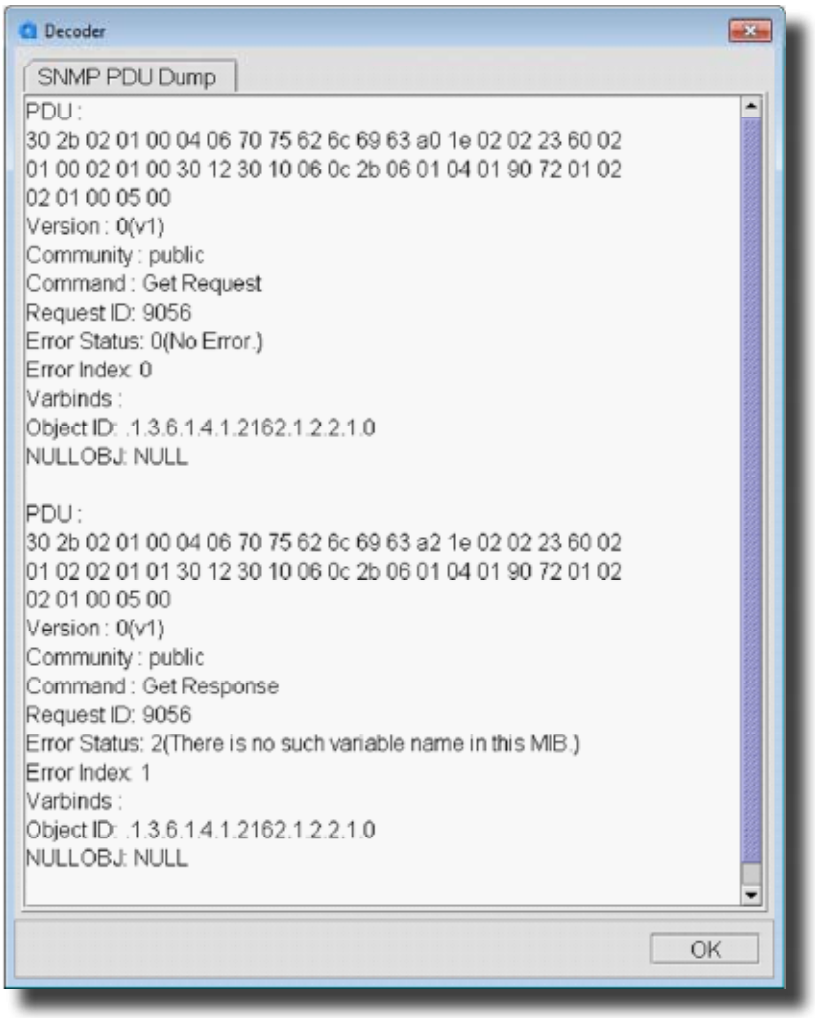
```
Check for noError [ErrorStatus, 0] by sending re-
quest to an accessible Scalar OID. Correct instance
[instance 0] value to be taken for testing.

Input :  OIDs Used : .1.3.6.1.4.1.2162.4.6.3

In the response varbind,Error Index should be 0 and
Error Status should be 0(noError).
```



The problem is the agent does not seem to support this MIB variable and for some reason WebNMS insists that it should be supported. This seemed odd to me but as I investigated I found that this OID is in the '2162' private enterprise number space registered to 'ZOHO' who is coincidently the manufacturer of this tool. In any case I can't understand why the tool would believe/insist that this MIB object must exist.

Below is the PDU decode window from this test. Notice that the agent seems to be properly reporting that there is noSuchName supported, but the tool insists that this is a CRITICAL error.

WebNMS PDU Decode Window showing the agent responded correctly with a noSuchName error code indicating that the reported bug is really a false positive.



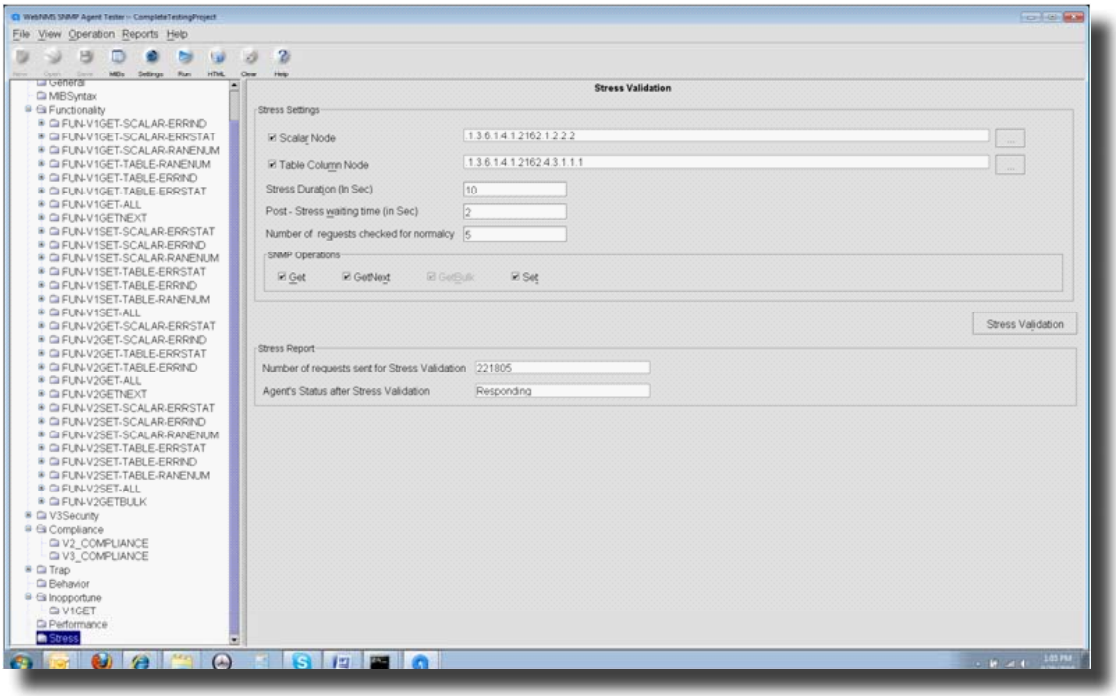
In all there were 16 Functional Test Errors reported, and all were attributable to the tool insisting on running various tests against the ZOHO private MIB.

Problems with WebNMS 'Stress' Testing

Moving on from functionality testing, I was intrigued to see what else WebNMS could do. Although the tool does some stress testing, this testing seems to be minimal and it unfortunately found no bugs in the agent even though other subsequent testing (with a different tool) later uncovered problems.

Notice that this test once again operates against the ZOHO private MIB by default.

Given all the issues I found related to the private ZOHO MIB I went back to manually 'unload' all MIBs except for MIB-II to see if these tests would run more smoothly. Even after making this change the tests still insisted on accessing the 1.3.6.1.4.1.2162 private MIB. I find this behavior quite baffling and I pity any test engineer with little to no low-level SNMP experience trying to figure out why these tests are failing.



WebNMS Stress Test Window which insisted on performing tests against ZOHO private MIB objects even though these were not supported by the agent.

Problems with WebNMS 'Behavior' Test

Next I tested WebNMS' Behavior Test feature which turned out to included just a single test. When run, this test reported a MAJOR failure in my testing. Upon investigation all I could find was a window with a blank description that popped up upon double clicking the failed test:

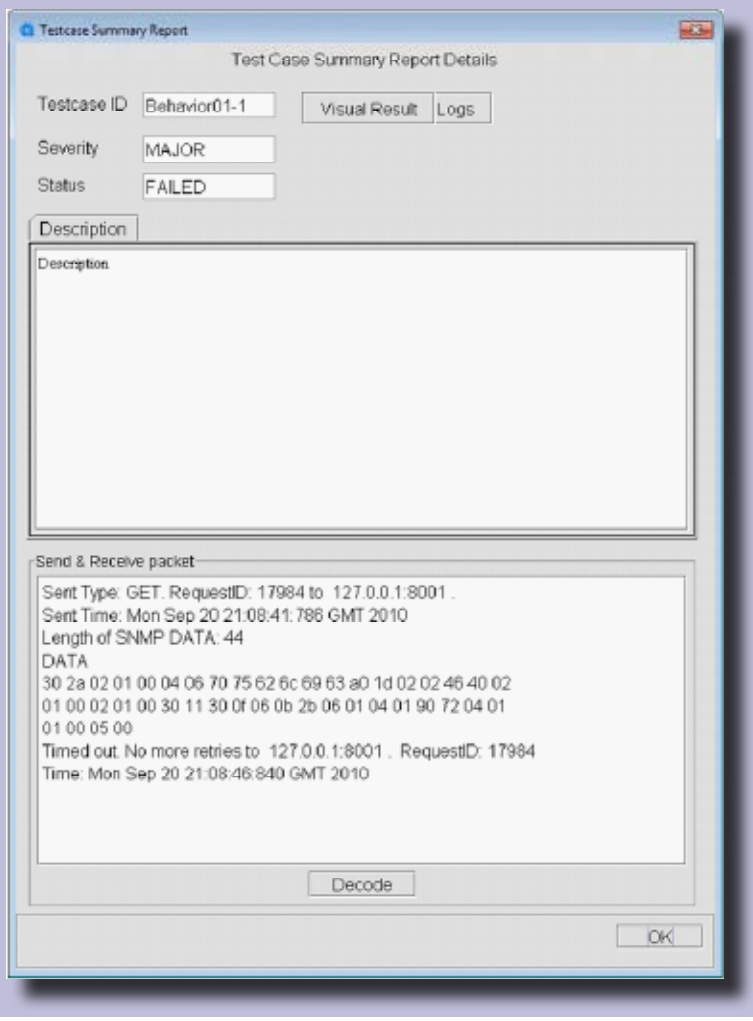
WebNMS Behavior Testing Problems

There are a number of problems with WebNMS' Behavior test report/summary:

1. The description is blank. I had no way to know what the test was trying to accomplish.

2. The test reports that it's sending test data to 127.0.0.1 (localhost) rather than the IP address of the agent I have configured (192.168.2.1). Since the test seems to have received no reply it probably is sending to a bad IP address different than the agent address used in all the other tests.

3. The test is operating on objects in the ZOHO private MIB (OID (1.3.6.1.4.1.2162.4.1.1.0), an object not supported by the agent.



Summary Findings for WebNMS

While WebNMS appears to be one of the few tools expressly designed to test (rather than simply monitor or manage) an SNMP agent, the tool has several shortcomings and limited functionality.

1. Out of the box it defaults a significant portion of its testing against the objects in the ZOHO private MIB. This caused tests to fail and required a significant amount of investigative research to isolate what was going wrong.

2. As for specialized MIB testing, the tool only includes support for the Printer and UPS management MIBs. If you happen to be testing a device in one of these categories this tool might be worthy of your consideration (although I did not verify the accuracy or usefulness of these tests since my device did not fall into either of these limited categories).

3. Boundary condition testing seems minimal.

4. Stress testing seems minimal.

5. Behavior test seems broken and does not send to the configured agent IP address.

6. This tester failed to find critical problems including SNMP agent crashes that were found using other tools in this report.

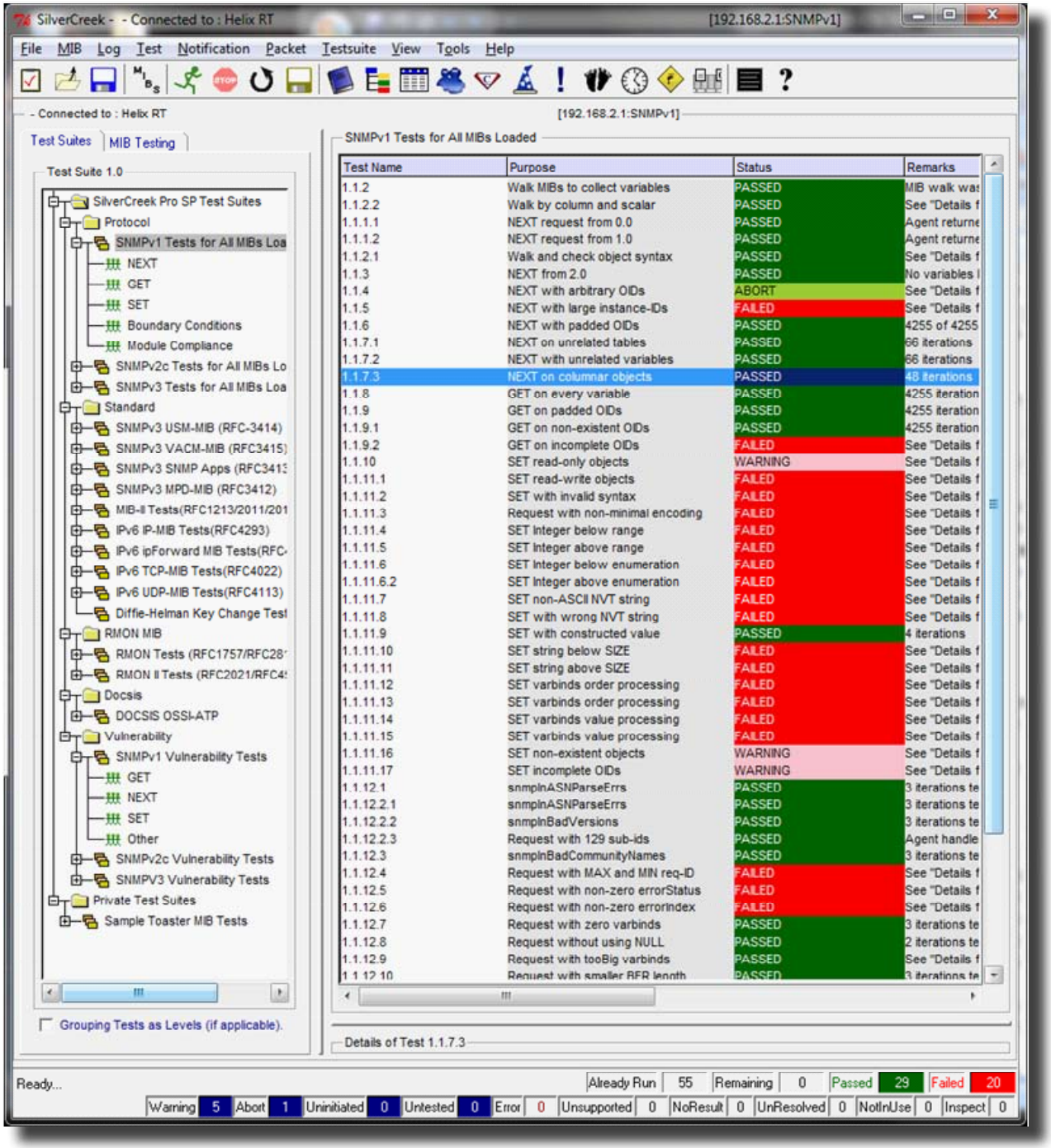
In summary I was very disappointed with WebNMS. It was difficult to get started and I found myself spending significant time diagnosing what the tool reported as CRITICAL and MAJOR errors that my research indicated are really problems with the tool.

InterWorking Labs SilverCreek

Installation of SilverCreek was straight forward although it did require that I obtain a license before I could begin testing. Once I installed the license the tool started very smoothly and I was able to start testing without the assistance of any documentation. The nicely organized user interface made it clear from the start that this was a serious tool intended to provide comprehensive SNMP testing of the device. The 'MIB walk' functionality that was largely the full extent of all but one of the other tools I tested was contained in just the very first of the 55 'protocol' tests provided.

SilverCreek Test Results

Unlike other tools, SilverCreek automatically loaded a full suite of standard MIBs. This saved me lots of time and hassle. After configuring the SNMP agent IP address and clicking 'Run', SilverCreek commenced running its battery of tests.



SilverCreek Identified Numerous Legitimate Agent Bugs

Almost immediately SilverCreek reported an error. Test 1.1.4 (NEXT with arbitrary OIDs) failed, the test description reading:

```
1.1.4   The purpose of this test is to verify that
the agent can perform correct lexicographic ordering
with non-variable arguments.  This test is run on
each variable returned in test 1.1.2.

This test removes the rightmost sub-identifier each
iteration and issues a GET-NEXT from the result-
ing OID, repeating until only two sub-identifiers are
remaining.  For a variable such as 1.3.8.6.4, this
means a GET-NEXT from 1.3.8.6, 1.3.8 and 1.3.

The expected outcome is that the agent returns the
first variable greater than the argument.

Reference    RFC 1157 § 4.1.3
```

And SilverCreek's log read:

```
No.  0  timeout 10
No.  1  timeout 10
Can't receive data from Socket, the agent is not en-
abled or may have crashed! Error ECONNRESET
```

After some investigation it turns out that the SNMP agent appeared to have completely crashed to the point where it was no longer responsive to any requests. Getting things working again required a hard reboot of the router being tested. After rebooting I continued by skipping the failed test in order to avoid repeating the crash.

This was the first example that was later repeated many times where SilverCreek found significant bugs in the agent that none of the other tools had identified. In all SilverCreek found a number of bugs in the SNMP agent device, including:

1. SNMP agent crash due to sending OIDs encoded with 32-bit numbers
2. Bugs in the '8072' private MIB returning values for non-existent OIDs
3. Failures with SNMP packets where the length was not minimally ASN.1 encoded
4. Failures with SNMP requests for extra large request-ids
5. Failures with SNMP requests with non-zero values in the errorStatus or errorIndex field(s)
6. Gaps in the MIB-II implementation were identified violating conformance guidelines

I came away feeling that SilverCreek had done a credible and thorough job testing the device. When tests failed, the built-in pop-ups explained each test's goals, procedures and rational. The test results were all clear and helpful. The tool seems robust, complete and well designed for testing SNMP agents.

Verifying Results with Additional SNMP Agents

After completing my testing of all the tools against a single common agent I decided to try running all the tools against a few additional agents just to see if the results would remain consistent across a variety of testing environments[3].

The results remained remarkably consistent across all the agents tested:

- The tools made for managing networks (rather than specifically testing SNMP agents) continued to find no faults at all.
- ZOHO WebNMS seems to have particular problems with unsupported MIB objects. Whenever an agent failed to implement particular objects WebNMS reports numerous errors even if the agent correctly returned the noSuchName error code as called for in the SNMP standards. This behavior generated massive numbers false positive bug reports.
- SilverCreek was the only product to find bugs in all the SNMP agents. SilverCreek found the most bugs and did so without generating false positives.

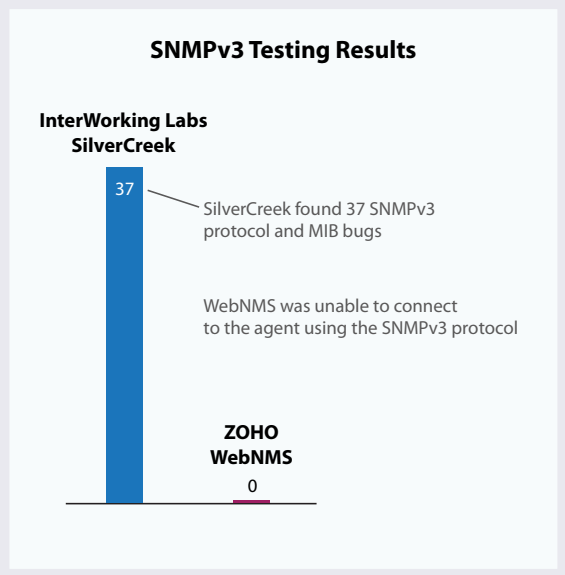
SNMPv3 Testing

As a last step I ran a suite of SNMPv3 testing. Since this more advanced version of the protocol was only supported by a subset of the agents, I couldn't use it exclusively to perform all the testing, but since the NetSNMP agent supported SNMPv3 I decided to round out the testing by using both SilverCreek and WebNMS to perform SNMPv3 validation.

After much trial and error I was unable to get WebNMS to successfully connect to the agent using the SNMPv3 protocol. Despite multiple attempts to get WebNMS configured as well as contacting their support team asking for help, I was never able to successfully perform SNMPv3 testing with this tool. Even if I had been able to get it to run it appears that all the 'functionality' testing is limited to the SNMPv1 protocol, so I'm not certain how much valuable SNMPv3 testing WebNMS could provide.

3 I tested all the tools against 3 SNMP agents:
1) Sierra Wireless AirLink Helix RT with 4.1.0.010 software version 4.1.0.010
2) Synaccess Networks netBooter series B, Part #1120 V2
3) NetSNMP, the default SNMP agent available for Linux distributions

SilverCreek connected to the agent using SNMPv3 without difficulty and ran a battery of SNMPv3 specific tests for both the SNMPv3 protocol and the SNMPv3 MIBs. SilverCreek found 24 additional SNMPv3 protocol bugs and 13 SNMPv3 MIB bugs for a total of 37 SNMPv3 related issues above and beyond all the bugs found in earlier testing.



I was impressed with SilverCreek's SNMPv3 support and the ease with which I was able to get this testing running. The implementation seemed thorough and complete.

Summary Results

SilverCreek was the only tool to find any legitimate bugs in all the tested SNMP agents. All other tools found no bugs at all or, in one case, identified many more false positives than legitimate bugs.

InterWorking Lab's SilverCreek found the most bugs and it did so without generating any false positives.

Testing with an 'SNMP Manager' Isn't Real Testing

Using an 'SNMP Manager' to do SNMP testing is clearly of little or no value. In my tests none of the SNMP Managers found even a single bug in any of the agents even though later testing found numerous critical issues.

It is clear that SilverCreek was built from the ground up to thoroughly test SNMP agents. Furthermore, it ran gracefully in a variety of agent environments automatically adjusting its behavior to the specific MIB objects implemented by each agent.

For developers and quality assurance engineers there is a clear and demonstrable difference between the various classes of test tool. Any organization wishing to ship a stable and reliable product would clearly be better off using a robust test suite like SilverCreek.

Real Testing Yields Real Results

The cost of licensing SilverCreek would in all likelihood be recouped many times over by resulting cost savings realized later in the product cycle. Finding bugs before a product gets in the hands of the customer saves both soft and hard costs. Product failures erode customer confidence as well as cost up to thirty-times more to fix in the field. Finding and fixing just one additional bug before a product ships could pay for the additional cost of using a premium tool like SilverCreek.