Global Information Assurance Certification Paper

Simple Network Management Protocol: Now More than a "Default" Vulnerability

SANS GCIH Practical Assignment
Version 2.0
Option 1 – Exploit in Action

Daniel Fluharty
March 22, 2002

Table of Contents

Introduction

Up until February 12, 2002, the primary focus on the open standard Simple Network Management Protocol version 1 (SNMPv1) with respect to vulnerabilities was the weak authentication, privacy and access control mechanisms (i.e., unencrypted community strings) provided by the protocol, compounded by the typical default (and thus widely known) community strings of "public" and "private." Ranked among the SANS Twenty Most Critical Internet Security Vulnerabilities (see http://www.sans.org/top20.htm), these deficiencies allowed attackers to remotely reconfigure or turn off network devices in a relatively easy fashion for exposed targets (i.e., those running the default or blank SNMP community names, or other easily guessable identifiers).

On February 12, 2002 at 2:30 PM EST, SANS briefed its community on a new set of widespread SNMPv1 vulnerabilities. It seems that these vulnerabilities were the result of programming bugs that had been in the SNMP implementations for quite some time, yet had only been recently found through research conducted by Finland's Oulu University Secure Programming Group (OUSPG, see http://www.ee.oulu.fi/research/ouspg, and http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/index.html for the paper). A multitude of targets are open to attack, including various network connection devices (e.g., routers, switches, bridges, WLAN access points), firewalls, antivirus appliances, networked printers, network management systems, operating systems, and any other component that utilizes the known vulnerabilities of SNMPv1. Due to the potential serious impacts of denial of service and compromise to millions of systems, analyzing and addressing these vulnerabilities should be of paramount importance to numerous organizations.

This paper gives an overview of these vulnerabilities, how attacks could be executed against them, how to protect against such exploits, and a recommended incident handling process that could be implemented to provide an adequate response. With the exception of the above brief discussion on the authentication issue, this paper will be limited to the vulnerability set identified by OUSPG. Further, reflective of the research conducted by OUSPG, this paper will be constrained to a discussion of these vulnerabilities within SNMPv1. It is important to note that a substantial amount of the content of this paper reflects a compilation of various sources, which have been noted throughout the text and/or reference list as appropriate.

Part I – The Exploit

Identification

The SNMP vulnerabilities discovered by OUSPG have been titled "Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)" by CERT/CC (CERT Advisory CA-2002-03, http://www.cert.org/advisories/CA-2002-03.html), and assigned (as of the date of this paper) Common Vulnerabilities and Exposures (CVE) candidate numbers CAN-2002-0012 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012) and CAN-2002-0013 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013) by MITRE. CAN-2002-0012 refers to the vulnerabilities in SNMPv1 trap handling; CAN-2002-0013 groups the weaknesses discovered in SNPMv1 request handling.

Brief Description

The researchers at OUSPG found a number of SNMPv1 vulnerabilities as part of their PROTOS - Security Testing of Protocol Implementations Project. Specifically, tests were conducted using the PROTOS c06-snmpv1 test suite (see http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/0100.html) against a range of widely-used SNMPv1-enabled products. The vulnerabilities can be grouped into two sets: those related to trap handling and those related to request handling. The impact associated with an exploit of these vulnerabilities ranges from unstable behavior to denial-of-service, as well as unauthorized privilege access in some cases.

The first set of vulnerabilities center on the manner in which SNMP managers decode and process trap messages (data used to report the status of an agent, including error or warning notices) sent from SNMP agents. The second set of vulnerabilities are due to the way in which SNMP agents decode and process SNMP request messages (data used to query an agent or to direct configuration modifications of a host device), sent from SNMP managers.

For a more complete description of the vulnerability, please see the discussion in part two.

Variants

These two sets of SNMP vulnerabilities apply to the SNMPv1 protocol itself; hence there are no known variants at the time this paper was published. Variance in these vulnerabilities may be discovered specific to products if the standard SNMPv1 protocol was not used in that product (i.e., the SNMPv1 protocol was modified to perform some custom function for a given product), or in a way that a specific operating system handles SNMPv1 messages.

Operating Systems

As a popular open standard protocol, SNMPv1 is supported in popular operating systems, including various Unix and Linux distributions, Microsoft Windows, Novell Netware, and Cisco Internetwork Operating System (IOS) products. Hence, SNMPv1 is not constrained to a particular operating system, or operating system version; it is widespread in modern, popular operating systems.

A partial list of popular operating systems implementing SNMPv1 follows:

- Caldera SCO OpenServer 5,
- Caldera UnixWare 7
- Caldera Open UNIX 8
- Cisco IOS (all major release trains, including 10.3 and earlier, 11.0, 11.1, 11.2, 11.3, 12.0, 12.1, and 12.2 releases)
- Compaq Tru64 Unix
- IBM AIX prior to 4.3, 4.3.x prior to level 4.3.3.51, and 5.1 prior to level 5.1.0.10
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows 98SE
- Microsoft Windows NT 4.0
- Microsoft Windows NT 4.0 Server, Terminal Server Edition
- Microsoft Windows 2000
- Microsoft Windows XP
- Novell NetWare 4.x, 5.x and 6.0
- Red Hat Linux 6.2, 7.0, 7.1, 7.2
- SGI IRIX 5.2-6.5

For more information with respect to a particular operating system (including patches or other fixes), please see the CERT advisory for links to vendor sites.

In addition to being integrated into operating systems, SNMPv1 is also implemented in the firmware of a number of components. The set of elements affected by these SNMPv1 vulnerabilities therefore includes:

- servers and workstations
- hubs, routers, switches, bridges, wireless access point components
- cable and DSL modems
- cameras and scanners
- networked printers, copiers, FAX machines
- network and systems management and diagnostic tools (e.g., network sniffers and analyzers)
- uninterruptible power supplies
- networked medical equipment (e.g., imaging machines and oscilloscopes)
- various manufacturing and processing equipment

(CERT CC bulletin CA-2002-03)

Protocols/Services/Applications

As mentioned earlier, the protocol affected is SNMPv1. Please see part two for a discussion of the protocol itself.

Some popular applications that implement SNMPv1 are also affected. A partial listing of these follows:

- HP OpenView Network Node Manager and Agent software
- iPlanet Directory Server 5.0, 5.0SP1 and 5.1
- iPlanet Web Proxy Server 3.6
- Netscape Directory Server 4.12, 4.13, 4.14, 4.15 and 4.16
- Sun Soltice Enterprise Agents
- Tivoli NetView for OS/390 Version 1 Release 2, 3 and 4
- Tivoli NetView for Unix Version 7.1and earlier
- Tivoli NetView for Windows Version 7.1 and earlier
- Tivoli Enterprise Console (SNMP adapter only)
- Tivoli Storage Network Manager

For more information with respect to a particular application (including patches or other fixes), please see the CERT advisory for links to vendor sites.

References

The following are useful references specific to these SNMPv1 vulnerabilities.

CERT/CC Advisory	http://www.cert.org/advisories/CA-2002-03.html
MITRE CVE Listings	http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013
OUSPG SNMP Discussion	http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/index.html Note: this page contains the PROTOS tool downloadable files that could be used by the attacker community to develop exploit tools.
SANS SNMP Tool	http://www.sans.org/snmp/tool.php Note: the SANS tool to locate hosts running SNMP may be requested from this page.

Part II – The Attack

Network Description and Diagram

The following network description and diagram are an example of one of an infinite number (since the protocol is so widely used) of configurations that these SNMPv1 vulnerability sets could affect. Please note that (with the exception of the workstation given as connected to the internal network) this example network architecture concept, description excerpts/concepts and diagram (please see figure below) are from the excellent GCFW paper by Brent Deterding, available online at http://www.giac.org/practical/Brent_Deterding.doc.

The following network description is largely a quotation from Brent Deterding's work, with minor additions and revisions as applicable to this assignment.

> **Overview:** This is a switched Ethernet network using gigabit routers, switched 10/100 to the desktop. The routers are Cisco routers running the latest stable version of IOS. It should be noted that in this example network (though not recommended), SNMPv1 traffic is permitted (e.g. by allowing traffic on 161/tcp and 161/udp, 162/tcp and 162/udp, or another specific protocol as dictated by a given product) from both external and internal hosts.

Test Machine: The test machine is a Linux box which is typically completely locked down running no services save SSH and denying all traffic but port 22 from the security administrator's IP. SNMPv1 is not running on the test machine.

External Router:
The external router will provide initial noise filtering by blocking all spoofed packets, any localhost packets (127), all reserved IP space in RFC 1918, and any source-routed packets. NetBios (tcp/udp ports 135-139) will also be blocked, as it is noisy and simply clogs up logs. SNMPv1 agent software is running on the external router.

External Firewall:
The external firewall is a Nokia IP330 box running Checkpoint Firewall-1 Enterprise. SNMPv1 agent software is running on the external firewall.

External Screened Subnet:

The screened subnet hosts several external servers. They are:

External Web Server: A large server with RAID 0 (striping without parity), it runs apache web server and receives any external HTTP and SSL (destination ports 80 and 443) traffic. SNMPv1 agent software is running on the external web server.

External DNS Servers: Servers (primary and secondary) running the latest stable version of BIND, they receive any external DNS (destination TCP/UDP port 53) requests. SNMPv1 agent software is running on the external DNS servers.

External Mail Server: A large server with RAID 0 (striping without parity), it runs the latest stable version of sendmail and receives any internal SMTP and POP-3 (destination ports 25 and 110) traffic going to or coming from the internal network and communicates with the internal mail server. SNMPv1 agent software is running on the external mail server.

External FTP Server: A large server with RAID 0 (striping without parity), it runs ncftpd and receives any external FTP (destination port 21) traffic and only uses passive-mode ftp. SNMPv1 agent software is running on the external FTP server.

Internal Router: The internal router also blocks all spoofed packets, any localhost packets (127), all reserved IP space in RFC 1918, and any

source-routed packets using the above-mentioned filters. SNMPv1 agent software is running on the internal router.

Internal Firewall:
The internal firewall is a Nokia IP330 box running Firewall-1 Enterprise. SNMPv1 agent software is running on the internal firewall.

Internal Screened Subnet:

DB Server: A large server with RAID 0 (striping without parity), it runs Oracle 8 and handles any database functions for the internal network, as well as requests from the external web server. SNMPv1 agent software is running on the DB server.

SSO Server: A server running Kerberos 4, it receives any Kerberos traffic (destination TCP/UDP port 88) and provides authentication to all hosts mentioned specifically in this document. SNMPv1 agent software is running on the SSO server.

DHCP Server: A server running the latest stable version of dhcpd, it receives any DHCP traffic (destination UDP port 68) and dynamically provides all internal hosts with real IP addresses. SNMPv1 agent software is running on the DHCP server.

Internal DNS Servers: Servers (primary and secondary) running the latest stable version of BIND, they receive any internal DNS (destination TCP/UDP port 53) requests. SNMPv1 agent software is running on the internal DNS servers.

Backup Server: A large server with RAID 0 (striping without parity), it performs backups using dump and restore. SNMPv1 agent software is running on the backup server.

Internal FTP Server: A large server with RAID 0 (striping without parity), it runs ncftpd and receives any internal FTP (port 21) traffic and only uses passive-mode ftp. SNMPv1 agent software is running on the internal FTP server.

Internal Mail Server: A large server with RAID 0 (striping without parity), it runs the latest stable version of sendmail and receives any internal SMTP and POP-3 (ports 25 and 110) traffic going to or coming from the external network, as well as any internal mail. For external mail, it communicates with the external mail server. SNMPv1 agent software is running on the internal mail server.

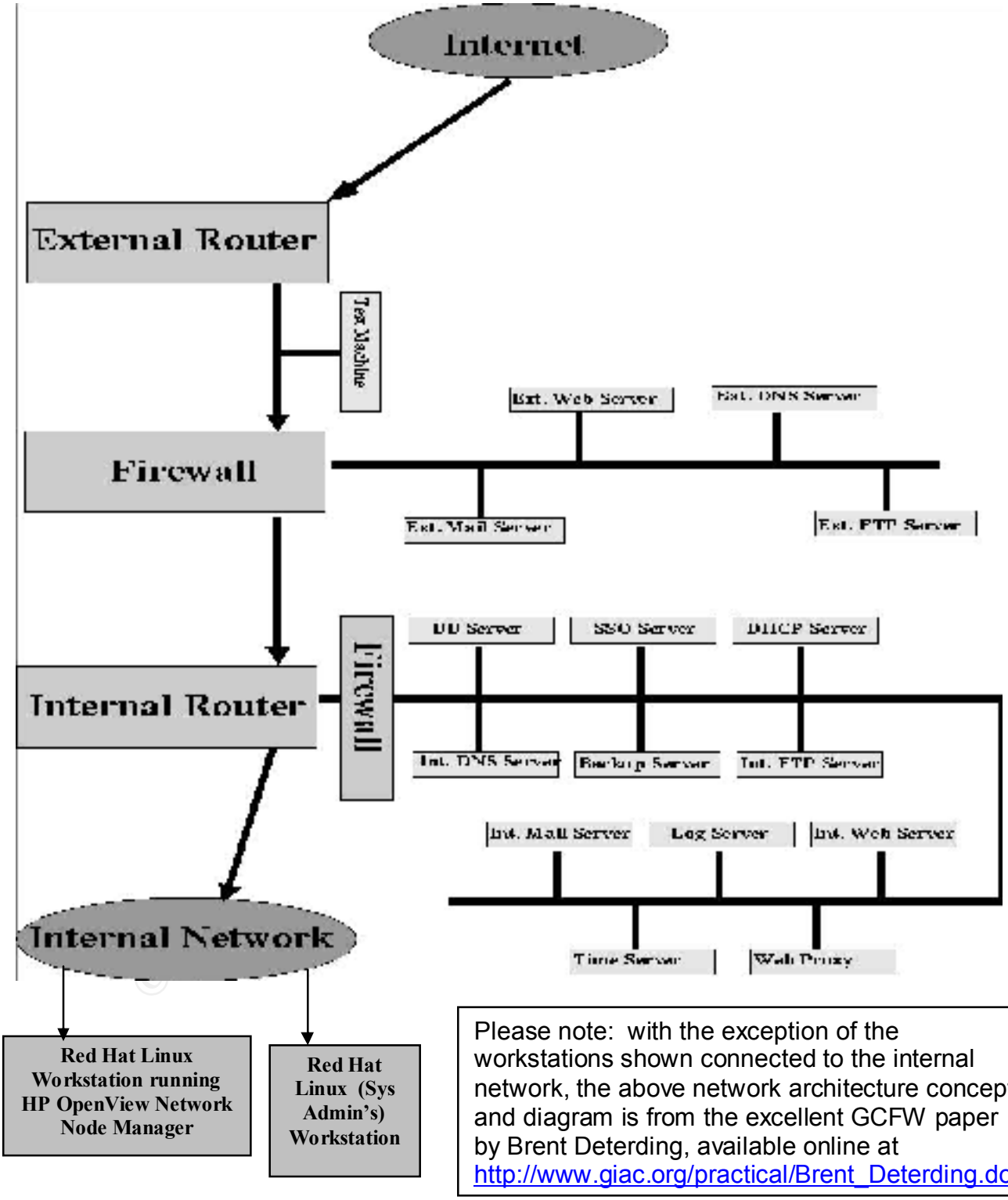
Log Server: A large server with RAID 0 (striping without parity), it runs the Modular syslog server and receives logs on port 514, which is tunnelled through SSH. SNMPv1 agent software is running on the log server.

Internal Web Server: A large server with RAID 0 (striping without parity), it runs apache web server and receives any internal HTTP and SSL (ports 80 and 443) traffic. SNMPv1 agent software is running on the internal web server.

Web Proxy: A Squid proxy running on a large server with RAID 0 (striping without parity), it receives any http-request traffic from the internal network on port 3128 and caches it. SNMPv1 agent software is running on the web proxy.

Time Server: A server running NTP, it acts as the clock-synchronization point for all machines on the internal and external screened subnets. It synchronizes it's time via a modem to the Naval Observatory. SNMPv1 agent software is running on the time server.

Red Hat Linux Workstations:
Running the latest versions of Red Hat, one is dedicated as a SNMP manager station and runs HP OpenView Network Node Manager software (using the SNMPv1 protocol); the other box is the system administrator's local machine.



Please note: with the exception of the workstations shown connected to the internal network, the above network architecture concept and diagram is from the excellent GCFW paper by Brent Deterding, available online at http://www.giac.org/practical/Brent_Deterding.doc

Protocol Description

Since the vulnerabilities discussed in this paper are inherent within the SNMPv1 protocol itself, it is useful to understand the fundamentals of the protocol.

SNMPv1 was developed in 1988 for the purpose of managing (including configuring and monitoring) networked devices. It operates at the application layer of the OSI model, and runs over TCP or UDP. SNMP is most commonly run on UDP ports 161 (request and response messages) and 162 (trap messages); however, SNMP traffic also uses TCP port 199 (SNMP multiplexer for handling of subagents – becoming obsolete), TCP and UDP ports 391 (SynOptics SNMP relay), TCP port 705 (AgentX extensible SNMP agent – similar to the SNMP multiplexer approach in that a master agent listens on a certain port (e.g. UDP 161), and then passes requests to subagents such as AgentX listening on TCP 705), and TCP and UDP ports 1993 (used for older Cisco implementations of SNMP). SNMPv1 is implemented on a number of devices (e.g., networking devices such as bridges, routers, switches, or hubs, copiers, faxes, printers, and manufacturing equipment) as well as a number of popular operating systems (including various Linux and Unix distributions/flavors, as well as Microsoft Windows and Novell Netware products). The SNMP protocol is formally defined in RFC1157.

Central to SNMP is the concept of managers and agents. The manager is usually a host machine that is used to control and monitor a set of agent devices, typically routers (but as noted above, may include other components including host machines). Managers run the SNMP client program, while the agents run the SNMP server program. Managers may oversee multiple communities of agents. Agents maintain a Management Information Base (MIB), a database that is used by the agent software to maintain performance information of the device. The manager has access to the values in the agent MIB, and can retrieve defined object values from agents. These values are specified by SNMP using the Structure of Management Information (SMI) method of object naming, data type definition (using as a base Abstract Syntax Notation One (ASN.1) definitions), and encoding rules (using standard Basic Encoding Rules (BER)). In addition to this monitoring capability, the manager can perform specific configuration or other functions by inserting values into an agent MIB. For example, a manager could set the value of a router reboot counter to “0,” which would cause the router to reboot itself the next time it checked the counter value. Finally, SNMP server programs running on the agents can also inspect its MIB, and if there is a problem or abnormal condition send an alert to the manager via a “trap” message.

Specifically, the five message types defined by SNMP are:

a) GetRequest – sent from manager (client) to agent (server) to obtain the value of variable in the agent MIB
b) GetNextRequest – sent from manager to agent to obtain the value of a variable following the ObjectID in the message; can be used to traverse the values in a table
c) GetResponse – sent from agent to manager, contains response (i.e., value of variable(s) requested) to a GetRequest or GetNextRequest message
d) SetRequest – sent from manager to agent to store a value in a variable
e) Trap – sent from agent to manager to alert of an event

Each SNMP message contains the version number (e.g. "1" for SNMPv1) and a community name defining the password (which is transmitted in plaintext). If no password is defined, then "public" is used. Agents and managers check the community name and sending address pair to determine if the other party has the rights to perform the requested operation.

In addition, GetRequest, GetNextRequest, GetResponse, and SetResponse messages contain the following fields:

a) Request ID – used to match request and response messages sent by managers and agents
b) Error status – integer value; "0" (no error) in request messages; in response messages, the value may be "0," "1" (response is too large to fit in the message), "2" (no such variable exists), "3" (the value sent by the manager to be stored is not valid), "4" (the value is read only and cannot be modified), or "5" (general/other error)
c) Error index – offset value used by a manager to determine which variable caused an error
d) VarBindList – group of variables and corresponding values manager is retrieving or setting

Along with the version and community name fields, Trap messages contain the following fields:

a) Enterprise – ObjectID of software package creating the Trap
b) Agent address – IP address of agent generating the Trap
c) Trap type – integer value; "0" (agent booted – coldstart), "1" (agent rebooted – warmstart), "2" (an interface link has gone down), "3" (an interface link has come up), "4" (invalid community string), "5" (EGP router neighbor down), or "6" (other messages)
d) Specific code – vendor-specific, defines specific code if Trap type = 6
e) Time stamp – shows time since event occurred that caused the trap
f) VarBindList – variables and corresponding values related to a specific trap

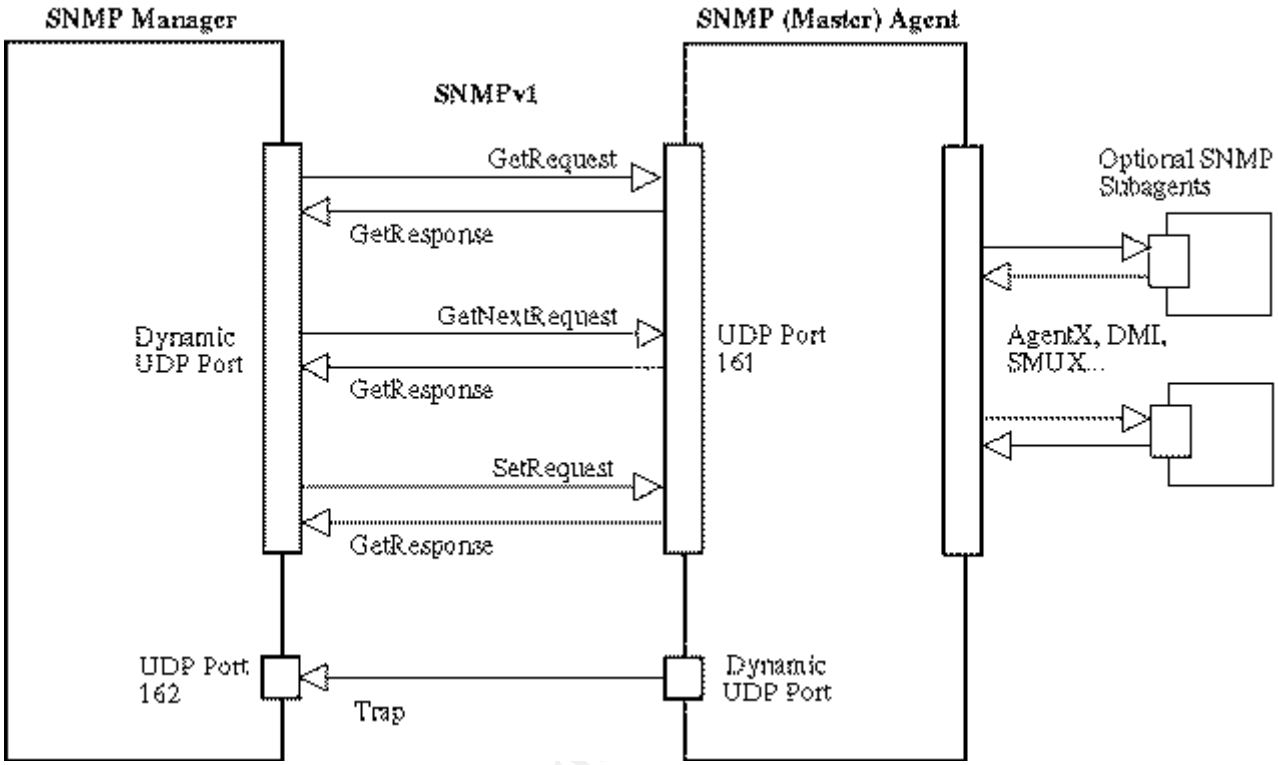
Due to it being a relatively simple protocol to implement, vendors can easily build SNMP agents to their products. Further, SNMP is extensible, allowing vendors to easily add network management functions to their existing products. Finally, commercial managers exist with GUIs and other add-ons, such as HP Openview Network Node Manager.

SNMP traffic example:

a) SNMP agent (server) listens on UDP port 161
b) A request is sent to the agent by the manager (client) from a dynamic UDP port to the agent UDP port 161
c) The SNMP agent receives the request, processes it, and sends a response back to the originating manager port
d) Also, asynchronous trap messages are sent by an agent using a dynamic UDP port to manager UDP port 162

Important notes:
SNMP message size is limited by maximum UDP message size.
The SNMP standard requires implementations to receive packets at least 484 bytes in length.

The following diagram is showing the SNMPv1 message architecture is from the OUSPG research paper, available at http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/index.html.



How the Exploit Works

First, it is important to point out that, as OUSPG notes, "SNMP agents and trap-aware SNMP managers are by design ready to accept incoming requests and traps without prior session setup" and hence are readily exploitable. The service simply listens on the designated port (typically UDP 161 and 162) and accepts arriving traffic. No state information is maintained, and hence rogue messages are readily accepted. Authentication is very weak, given the plaintext community string names that are used. Weak authentication is further compounded by the use of "public" and "private" as common default names.

OUSPG performed two sets of tests of SNMP request message handling: one test focused on ASN.1 decoding, the second looked for exceptions in the processing of the decoded data. The first set of vulnerabilities center on the manner in which SNMP managers decode and process trap messages (data used to report the status of an agent, including error or warning notices) sent from SNMP agents. The second set of vulnerabilities are due to the way in which SNMP agents decode and process SNMP request messages (data used to query an agent or to direct configuration modifications of a host device), sent from SNMP managers. Hence, as is evident from the above statements, both sets of

vulnerabilities are a result of the manner in which SNMP decodes and processes messages. SNMPv1 does not perform a check to see if messages are legal, or within the parameters of the protocol specification; rather, it accepts any input (including exceptional or erratic input) that has been correctly encoded by a BER encoder. This allows an attacker to issue malformed inputs that can cause an array of impacts, including denial of service and the execution of arbitrary commands. OUSPG's testing method was "black box" and hence specifics of the exploits inner workings of the SNMPv1 processing vulnerabilities are not yet published. Instead, a multitude of test data was sent to various implementations of SNMPv1 to see how exceptions were handled. OUSPG defines an exceptional element in the test cases performed as "a piece of data designed to provoke undesired behavior of the subject implementation" (http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/index.html). Further, OUSPG notes that any given test-case may contain multiple exceptional elements. While the exceptional elements they used did include clearly illegal messages, they noted that many of them were actually legal or "in the gray area" but not fully considered when building the software.

The specific exceptional element categories are as follows (from the OUSPG website above):

Exceptional element categories	
Name	**Description**
E-01	Invalid BER length (L) fields. The first part of this group contains valid encodings of exceptional values, while the other part contains invalid length of length BER encodings
E-02	Exceptional elements for Class tags: Universal, Application, Context-specific, Private
E-04	Exceptional elements for BER's T fields number field
E-05	Mostly invalid encodings for BER NULL type
E-06	Mostly invalid encodings for BER BOOLEAN type
E-07	Mostly invalid encodings for BER INTEGER type
E-08	Mostly invalid encodings for BER OCTETSTRING type
E-09	Mostly invalid encodings for BER BITSTRING type
E-10	Mostly invalid encodings for BER SEQUENCE-OF type
E-11	Mostly invalid encodings for BER SET-OF type
E-12	Mostly invalid encodings for BER OBJECT-IDENTIFIER type
E-13	Mostly invalid encodings for BER REAL type
E-14	Mostly invalid encodings for BER ENUMERATED type
E-15	Total garbage with semi correct BER encodings
E-16	BER elements of PDU totally removed one at a time
E-17	Invalid extra data inserted in different parts of the PDU
I-01	Overflows with multiple zeroes and integer coded format strings
I-02	Overflow integers: various very big integers from (+/-)1 to magnitudes (+/-)2^256 and above
I-03	Large boundaric integer values (ie. (2^32)+-1, (2^64)+-1,...)

Exceptional element categories	
Name	**Description**
I-04	SetRequest(3) and Trap(4) PDU types in SNMP messages are replaced with big integers (ie. (2^32)+3, (2^56)+3, (2^64)+3,...)
B-01	Legal values (0x01 - 0x05) for ErrorStatus defined in RFC1157
O-01	General overflow strings consisting of for example from long strings of character 'a'
O-02	Long exceptional strings applied after community name public as VLAN name (ie. public@vlan0). Strings include multiples of 'a', "%s", '0', etc.
O-03	Overflows with null terminator (0x00) inserted at their beginning
O-04	Overflows with null terminator (0x00) inserted in their midst
F-01	Format strings of type "%s", "%s%n%x", "%.999d", "*.*d" and their variants
F-02	Community-specific format strings applied with community name public, ie. "public%s"
O-05	Object-identifier exceptional elements. Very long OIDs, OIDs with extremely big (ie. negative) branches
O-06	Object identifiers for sys.sysDescr, sys.sysName, if.ifNumber if.ifIndex with no, or invalid indexes
O-07	multiple (empty) VarBind entries. OIDs used in VarBind entries are if.ifIndex for Trap-PDU's and sys.sysName for other PDU types
O-08	multiple VarBind entries with short format string ("%s%s%s") as value. OIDs used in VarBind entries are if.ifIndex for Trap-PDU's and sys.sysName for other PDU types.
M-01	Zero length (null) data applied as a value of every BER element of PDU

Legend:

- B: Bit pattern exceptions
- E: BER encoding exceptions
- F: Format string exceptions
- I: Integer value exceptions
- M: Missing symbol exceptions
- O: Overflow exceptions

OUSPG showed a number of failed test cases across a range of (unidentified to provide vendor anonymity) product implementations. Exploits for some of the implementations did not even require a correct community name be entered; the default community name of “public” was used to run the tests in the other cases (i.e., it is assumed that the string name had to be known for those cases to execute). The research group claims that each test case failure can be taken to mean that the vulnerability will allow for denial of service. Further, OUSPG’s findings show that many of the exploits (exceptions) used in their test suite corrupted memory or the stack, allowing for buffer overflow exploits to include the execution of arbitrary code and/or the modification of the attacked system (see http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/index.html). Relevant to the example used in this paper, Red Hat confirmed that the test-suite caused several failures (including denial of service, service degradation, and remote execution of arbitrary code) in the ucd-snmp tools in version 4.2.2 and

earlier (http://www.redhat.com/support/errata/RHSA-2001-163.html). Further, Cisco stated that these exploits caused denial of service conditions in numerous IOS and non-IOS products (please see the Operating System section for more information, as well as the advisory at http://www.cisco.com/warp/public/707/cisco-malformed-snmp-msgs-pub.shtml). These impacts will be assumed for the remainder of this paper. For a byte-by-byte examination of one of the OUSPG test cases, please see the Attack Signature section.

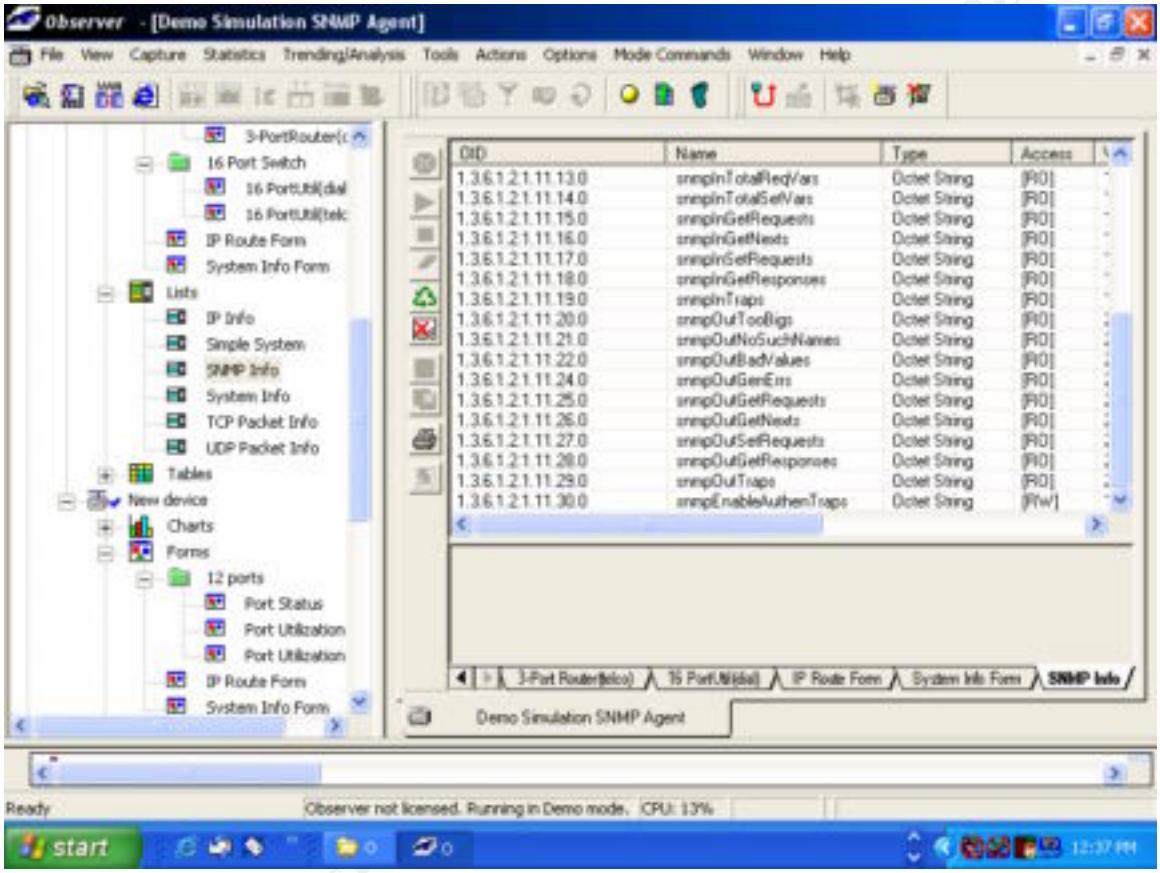
Attack Description and Diagram

In the sample network used for this paper, SNMP traffic is allowed to traverse into and out of the network. Given that the vulnerabilities found by OUSPG allow an attacker to cause a denial of service on SNMP components, or to execute arbitrary code on these components, a number of combinations of attacks are possible. The reader is reminded that execution of these exploits is not limited by the community name weak authentication measure. For one, a portion of the exploits do not need to know the community name at all (as reported by OUSPG). Also, even if the name were needed, the plaintext transmission of community strings provides no protection from outsider eavesdropping and later reuse.

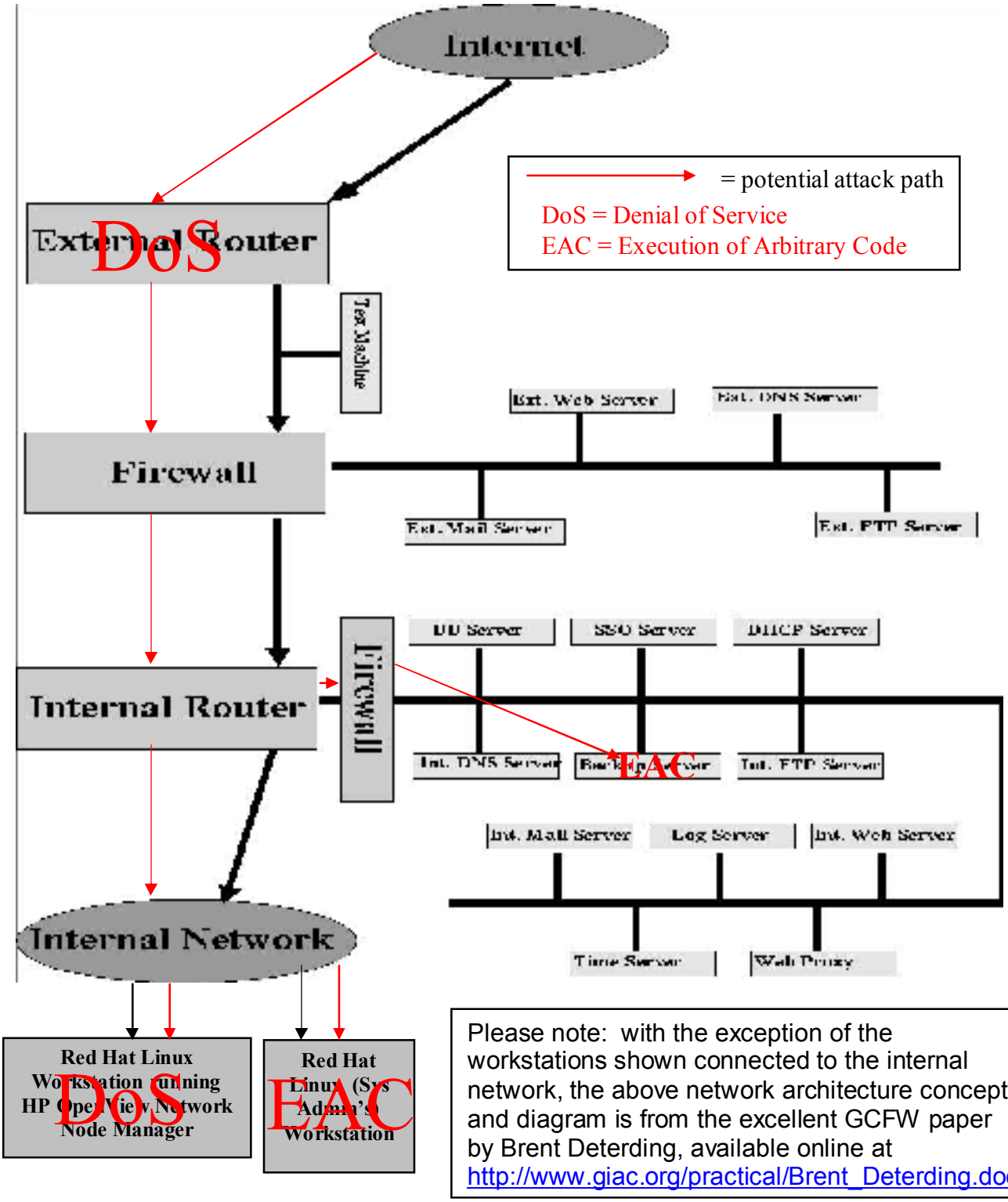
Given that the external Cisco router is vulnerable, an attacker from the Internet could use the OUSPG PROTOS test kit (or a refined derivative of it created specifically for executing successful attack scripts) to shut the router down, causing a denial of service to the entire organization's network with respect to the Internet connection. A denial of service attack may also be launched against the HP OpenView Network Node Manager to prevent the rightful management of SNMP agents (which could obviously be useful in an attack). For the Java archive (jar) files comprising PROTOS, see http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/index.html).

Further, since these vulnerabilities allow for the potential to execute arbitrary code on hosts running the SNMPv1 software, a number of additional existing exploits could be brought in to gain root or user level access on these machines. Sensitive machines, such as the backup server and system administrator's local machine may be potential targets in this example. Once compromised, these hosts may then be used as launch points to other targets, possibly exploiting trust relationships between the machines.

While denial of service or execution of arbitrary code can take various forms, a protocol analyzer such as Observer by Network Instruments could be used to capture packets and analyze messages, as depicted below:



The following updated network diagram shows several different potential attack paths and impacts with respect to the above discussion. Please note that a number of other possibilities exist due to SNMPv1 being extensively deployed; this is an example only.



Please note: with the exception of the workstations shown connected to the internal network, the above network architecture concept and diagram is from the excellent GCFW paper by Brent Deterding, available online at http://www.giac.org/practical/Brent_Deterding.doc

Attack Signature

The attack signature could take a number of different forms. One could be simply SNMP traffic from an unrecognized IP address. More practically and specifically, a signature could be a large number of SNMP GetRequest messages sent in a relatively short duration of time to a common SNMP port, such as UDP 161, on a range of hosts. This may signal the scanning for listening SNMP agents. Scanning activity may also be seen in the form of trap messages sent to hosts, looking for managers. Another attack signature could be multiple successful or unsuccessful attempts to modify MIB values in a short timeframe.

Since the payload is carried in the SNMP message itself, an application-level proxy firewall or an intrusion detection system could inspect the packets for malformed, or exceptional elements, such as those found by OUSPG. Based on OUSPG's research, there are conceivably an infinite number of possibilities for malformed packets that could cause a denial of service condition or buffer overflows. For instance, some of the SNMP payload test cases generated by OUSPG included invalid BER encoding formats, multiple empty varBind entries, overflows created by long strings, and very long object identifiers (OIDs). Since most of these exploits depend on invalid encoding or extra "junk" being inserted into a valid SNMP packet, there is no finite set of signatures to consider (apart from the set of test cases released already by OUSPG that are known to cause error conditions on different implementations).

An example of a malformed SNMP message (viewed ala the Hackman tool from TechnoLogismiki) follows. It is OUSPG test case #00000200.

However, before examining this data, it is useful to review the BER coding standards used in SNMP. Each piece of data is to be encoded in triplet format consisting of tag, length and value. The following summary table of data types is derived from Forouzan's *TCP/IP Protocol Suite*:

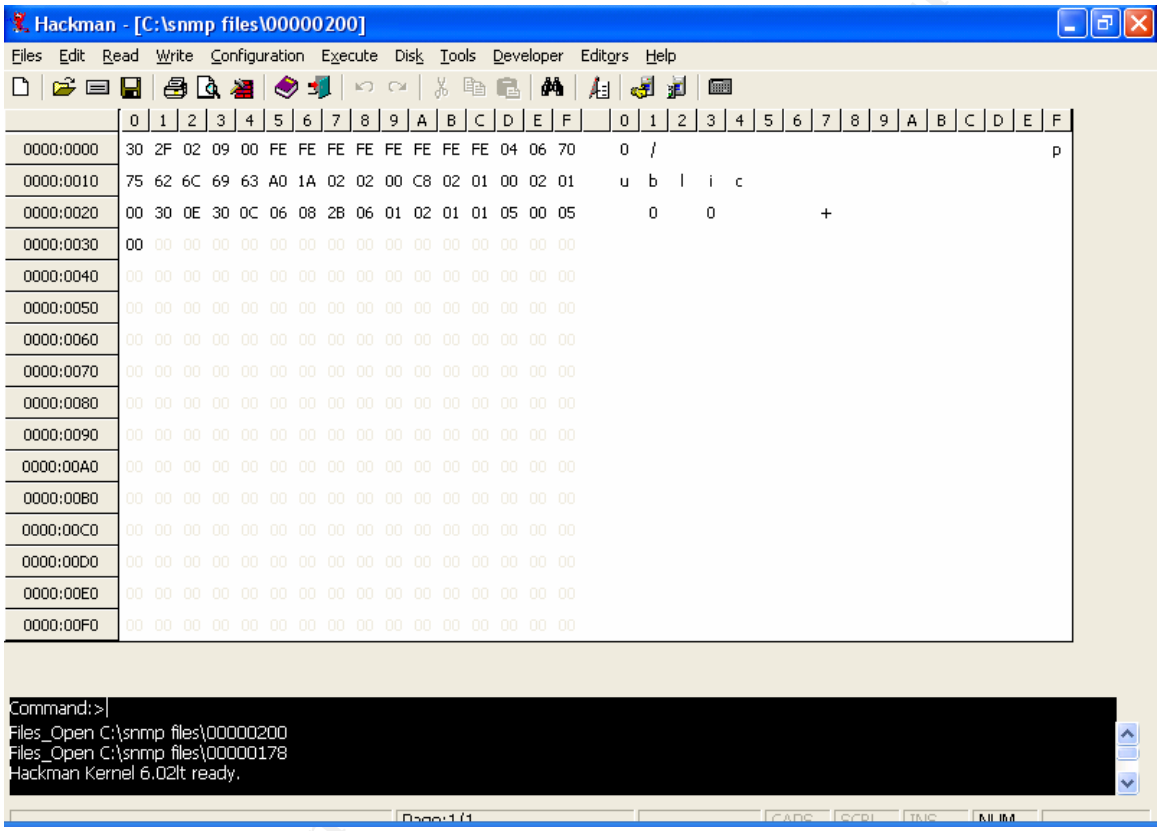
Data/Message Type	Binary Tag	Hexadecimal Tag
Integer	00000010	02
String	00000100	04
ObjectIdentifier (OID)	00000110	06
Sequence, sequence of	00110000	30
IPAddress	01000000	40
Counter	01000001	41
Guage	01000010	42
TimeTicks	01000011	43

GetRequest	10100000	A0
GetNextRequest	10100001	A1
GetResponse	10100010	A2
SetRequest	10100011	A3
Trap	10100100	A4

So, the payload shown in Hackman can be interpreted as follows:

Byte Location	Content	Explanation/Comments Green = Normal; Red = Abnormal
0000-0001	30 2F	Denotes the total sequence length $2F_{16}$ or 47_{10} bytes; note: byte length does not include these first two bytes, so the length given is actually correct in this case
0002-0003	02 09	Denotes integer of length 09_{16} or 9_{10} bytes (supposed to be used to identify the SNMP version number; usually only 1 byte versus 9)
0004-000C	00 FE FE FE FE FE FE FE	Gives the version number as $00FEFEFEFEFEFEFE_{16}$ or 71775015237779198_{10} – clearly wrong
000D-000E	04 06	Denotes a string of length 06_{16} or 6_{10} bytes (to be used for the community string name)
000F-0014	70 75 62 6C 69 63	Gives the community string name as “public”
0015-0016	A0	Denotes a GetRequest of length $1A_{16}$ or 26_{10} bytes
0017-001A	02 02 00 C8	Denotes an integer of length $02_{16} = 2_{10}$ bytes, Request ID = $00C8_{16}$
001B-001D	02 01 00	Denotes an integer of length $01_{16} = 1_{10}$ bytes, Error Status = 00 (note this is correct, as error status is 0 in request messages)
001E-0020	02 01 00	Denotes an integer of length $01_{16} = 1_{10}$ bytes, Error Index = 00 (again, this is appropriate)
0021-0022	30 0E	Sequence of length $0E_{16} = 14_{10}$ bytes
0023-0024	30 0C	Sequence of length $0C_{16} = 12_{10}$ bytes
0025-002E	06 08 2B 06 01 02 01 01 05 00	ObjectID of length $08_{16} = 8_{10}$ bytes, udpInDatagram
002F-0030	05 00	Null entity of length $00_{16} = 0_{10}$ bytes

It is interesting to note that the only (albeit significant) anomaly in this instance is that the version number is unusually large (it is test category I-02, “Overflow integers: various very big integers from (+/-)1 to magnitudes (+/-)2^256 and above.”) An instance such as this is one of the conditions that have been shown to cause denial of service conditions in some implementations of SNMPv1. As noted before, there are many other possible variations.



Countermeasures

Since these vulnerabilities are found in the SNMPv1 protocol itself, many products are affected. Vendors including SNMPv1 in their products must analyze their particular implementations, and construct patches for SNMP software that addresses the vulnerabilities by performing checks on messages for conformity with the protocol, and providing handling routines to deal with unexpected inputs. A number of vendors have already released such patches (see http://www.cert.org/advisories/CA-2002-03.html#vendors for an updated list of vendor patches and recommendations).

Persons affected by these vulnerabilities can take several steps to remedy the situation.

1. Determine the devices on your network that are running SNMPv1. Tools are available to assist in this task, such as:
   - SNMP Tool (SANS) – http://www.sans.org/snmp/tool.php
   - SNScan (Foundstone) – http://www.foundstone.com/knowledge/free_tools.html

2. If possible, remove or disable SNMP services to the extent possible for your environment. Consider using another means to manage your network. Note: even by disabling SNMP, some of the OUSPG tests were able to exploit the vulnerabilities mentioned.

3. Apply vendor patches – see the above URL for a list of available patches.

4. Perform ingress filtering of incoming port 161/udp (SNMP), 162/udp (SNMP system management messaging), 199/tcp, udp (SNMP Unix multiplexer), 391/tcp, udp (SynOptics SNMP relay), 705/tcp (AgentX), and 1993/tcp, udp (Cisco SNMP).

   For example, the following configure commands could be used on Cisco router access control lists:
   ```
   access-list 100 deny udp any any eq snmp
   access-list 100 deny udp any any eq snmptrap
   access-list 100 deny tcp any any eq 199
   access-list 100 deny udp any any eq 199
   access-list 100 deny tcp any any eq 391
   access-list 100 deny udp any any eq 391
   access-list 100 deny tcp any any eq 705
   access-list 100 deny tcp any any eq 1993
   access-list 100 deny udp any any eq 1993
   ```

5. Perform egress filtering on the above ports to prevent your internal network from being a source of attacks.

6. Of course, ensure the default community string names have been changed.

Part III – The Incident Handling Process

Continuing in the hypothetical line of thought behind this paper, the following is an incident handling process that could be used to address the exploit of these SNMPv1 vulnerabilities.

Preparation

Countermeasures

In our sample network (which is typical of numerous implementations in practice), SNMPv1 was running on a number of devices. While there were routers and firewalls in place, these devices did not filter or otherwise regulate SNMP traffic, either flowing into or out of the network. There were no network- or host-based intrusion detection systems installed.

Despite the deficiencies outlined above, this organization had a number of countermeasures and other controls in place. Specific to SNMP, all community string names had been changed from their defaults. Further, the organization had established policies and procedures to facilitate information confidentiality, integrity and availability goals. Policies outlined system user roles and associated privileges, as well as authentication, authorization, access control, object reuse and auditing policies. Further, the policies outlined guidelines stating the particular controls, such as encryption or integrity checks, specifically needed for certain information sets. These policies were flowed down into simple, concise procedure sets to ensure the policies were enacted appropriately and correctly.

Example excerpts of relevant policies include:

- All employees, contractors, and other persons having access to the organization's information systems shall immediately report any and all suspected and actual breaches of information security to the information security manager (Mr. John Doe; 909-555-1234; room 123) via phone or in person. If the information security manager is unavailable, contact the organization help desk at 909-555-4321 (room 321). The help desk will contact an incident response team member as appropriate.
- The information security manager or designee is responsible for determining the appropriate level of response to a security incident.. Senior management will be kept apprised of all developments impacting the mission of the organization or its resources.
- All employees shall be trained regarding the appropriate reporting of security breaches.

- All employees, contractors, and other persons shall only be granted access to that information to which they are authorized according to their need to know in order to perform their duties.

A summary of other policies is also embedded in the following paragraph.

This organization had relatively strong physical security controls in the form of access-controlled entry points, locked equipment and media spaces and containers, close-circuit television monitoring, and a physical security staff to perform checks, escort visitors, and oversee other aspects of the organization's physical security. Persons holding sensitive positions were required to undergo background investigations to establish credibility. Users were granted access to information resources on a strict need-to-know, least-privilege basis. Separation of duty controls were also employed. User accounts were managed in a secure fashion, by closing inactive accounts immediately, password-cracking programs to ensure adequate passwords were used, and requiring users to complete security training prior to being granted system or network access.
The information security staff at the organization consisted of an information security manager (who had other duties) and a system administrator (who performed much of the technical security support services). This staff did conduct periodic audits to ensure the policies and procedures documented were carried out by the organization. They also maintained a reasonably accurate inventory of the organization's information assets, including hardware, software, and data sets. In addition to the information security manager and the system administrator, there were a couple additional personnel selected for the incident response (IR) team. One was another trusted system administrator within the organization, selected for her expertise in the systems the organization ran and her trustworthiness. The role of this SA in the IR team was to provide secondary support to the primary, augmenting his efforts as necessary. The organization's two security guards were also members of the IR team, providing physical security support and facilitating communications across the team and its interfaces.

Incident Handling Process

This organization had a formal incident handling process in place that was approved by management and aligned with the organization's mission. While the information security manager was the overseer of the incident handling "team," the system administrator identified above was responsible for working through most of the steps in the process. This person was well qualified in both security matters and system-specific knowledge. Incident categories were established as follows:

- Category 1 = Root/admin level compromise

- Category 2 = User level compromise
- Category 3 = Denial of service
- Category 4 = Malicious logic incident
- Category 5 = Scanning or probing activity
- Category 7 = Poor security practice
- Category 8 = Unauthorized use by an employee or other organization trusted "insider"

Checklists were created to be used in the event an incident should occur, and tailored to the incident category at hand. An emergency communication plan was included in the incident handling process, detailing a call tree that was dependent on incident type. For example, senior management was to be notified in the event of a root level compromise, but scanning and probing activity was handled at a lower level. As part of the user training, incident recognition and reporting procedures were included. Further, the incident handling team received recurring technical and process training specific to their role.

Identification

Given the attack scenarios described in part two, the incident could have been discovered by a lockup of the network (e.g., Denial of Service attack against the external router) or by a user who noticed changes to files (e.g., execution of arbitrary code resulting in a user- or root- level compromise). Once reported to the incident handling team, the system administrator would begin a log of activities performed to be used for later evaluation and possibly as a legal document. The SA would first determine that the incident reported was indeed an incident by reviewing the component(s) for simple errors or mistakes, such as a misconfiguration causing a denial of service or a user forgetting where a file was last stored. Upon confirmation that the suspected incident was indeed real, either as a denial of service or some consequence of the execution of any code (as this vulnerability permits in some cases), the handling process would continue.

Each piece of evidence would be clearly listed in the log, including dates, times, serial, make and/or model numbers, or other relevant identifiers. All notes and printouts would be chronologically numbered and signed by the handler. Access to evidence would be strictly controlled by locking it in secure containers, ensuring only the authorized persons have access, and that access is accounted for by logs or other means.

To give a specific example of identification, assume that multiple users have reported that they are no longer able to transmit or receive packets to or from the Internet. The system administrator goes to check the external router, thinking to

himself, “I have a suspicion this may be related to the SNMP vulnerabilities warning that SANS just released a couple of days ago.” “I’ve read the message and understand the vulnerabilities and fixes, but just haven’t gotten around to doing anything about it yet. I was going to do something later this week…surely no one’s exploited us on this already!”

Hoping it may be just a power or cable issue, the SA checks the power supply and cables to make sure all are intact and functioning. Having verified that the physical components are not faulty, the SA powers down the router, waits a minute, and restores power. Almost as soon as the router comes back up, it immediately begins to reboot. The SA disconnects the router from the Internet and moves on to check the router logs.

Unfortunately, to the SA’s disappointment, the router logs show continual, repeated reboots, and the router MIB (accessed using an HP OpenView browser connected to the router directly via a laptop connected to the router console port) shows the reboot counter to be “0,” indicating yet another reboot is scheduled the next time the router checks the MIB. Further, logs show that the reboots have been caused by the MIB reboot counter being set to “0” on continual, repeated occasions.

A sample of the log extracts follows:

```
02262002 09:01:03 reboot_counter set = 0
02262002 09:03:07 reboot_counter set = 0
02262002 09:05:11 reboot_counter set = 0
02262002 09:07:15 reboot_counter set = 0
02262002 09:09:19 reboot_counter set = 0
02262002 09:11:23 reboot_counter set = 0
```

“It has the characteristics of that SNMP vulnerability sure enough.” Since the events have caused the router to essentially go down, no packets have made it to other network components, and thus no further correlation can be performed at this point. The SA concludes that it is best to assume that he has a probable incident on his hands in the form of a denial of service; this decision is made to be on the safe side, and is based on the information he knows about the vulnerability, his lack of addressing it to date, the events logged on the router, and the denial of service condition that the events have caused). Even if it is not caused by this SNMP vulnerability set, the SA knows that it is best to contain the situation (given the lack of adequate countermeasures to these vulnerabilities), apply fixes, and test for further compromise or denial of service in other parts of the network. Once the incident response process is executed, at least this widespread vulnerability will have been addressed, and the posture of the

network analyzed and strengthened. He carefully follows the steps outlined above, and moves on to ensure the problem is contained.

Containment

Before doing anything further, the SA secures the immediate area housing the external router and firewall. He knows that this is an important step to ensure the integrity of the handling process is not compromised. The organization's physical security staff is directed to limit access to this area to only the incident handling team until further notice.

Knowing that his network is unprotected against these SNMP vulnerabilities, the system administrator knows it is best for now to leave the router disconnected from the external network, and to also disconnect it from the internal networks; however, the power is left connected to maintain the state of the machine. Next, to determine the extent of the problem, the SA checks the next logical device, the firewall connected directly to the external router. Any traffic destined to the organization's network components would first have to pass through that Checkpoint firewall. Examining the firewall logs, no unusual activity, including SNMP scans, were found. This fact, combined with the lack of reports of other internal system troubles, led the SA to believe that they were in a sense lucky this time. This was likely only a denial of service attack against the external router. "Probably some script kiddies who thought they'd point a new tool against any victim," he thought to himself.

The SA went back to inspect the external router more closely. He tool along his jump bag, containing a tape recorder to note comments for later review, a copy of the IOS binaries, backup utilities (including Safeback and Ghost), fresh backup media, forensic software, a small hub, his dual-OS laptop, cell phone, call list, and phone book.

Examining the logs again (see the above excerpt), it was clear that the router had been experiencing a round of DoS attacks. The router was continually rebooting itself. So (specific to this case and vulnerability), unless another attack were carried out earlier, the problem was contained to the external router. This DoS (since caught just after it started), and having disconnected the router, actually prevented further perhaps worse attacks (by the execution of arbitrary code which could lead to a root- or user-level compromise). The firewall didn't show any unusual SNMP traffic, and the SA concluded that this incident was limited to DoS against the external router.

Next, the SA connects his laptop to the router's console port using a serial cable (EIA/TIA-232 protocol). The Hilgraeve HyperTerminal terminal emulation

software is started to provide an interface to the router to perform a backup of the configuration file and IOS software in its present state. Then the SA connects another laptop with a clean, removable hard disk drive directly to the router to serve as the "tftp server" to backup the files.

The commands `copy running-config tftp` and `copy flash tftp` are used by the SA to create image backups of the configuration file and IOS software for inspection. Using Ghost, the SA next makes two binary (bit-by-bit) copies of these files onto clean CD-R disks using the CD burner in the tftp laptop. One of these copies is clearly marked with the date and time of creation, and signed by the SA. The SA then stores this copy in a signed and dated envelope, which is placed in a secure container.

The SA next conferences with the information security manager and the organization's senior management to relay his preliminary findings. The consensus is to remain disconnected from the Internet until the external router and other organizational assets can be secured against these SNMP vulnerabilities.

Eradication

Having isolated the attack, the SA next seeks to perform a vulnerability assessment first on the external router itself. He first compares the MD5 cryptographic checksums of the current IOS software against that of the software known to have absolute integrity (e.g., the CD-ROM shipped directly by Cisco). The checksums agree, and hence the SA confirms that the IOS software itself was not affected. Next, he analyzes the configuration file taken from the attacked router against the last known good backup of the router configuration file. These two also match, verifying that no rule sets or other settings were modified from the attack. As an added precaution, the SA takes time (with the help of another trusted system administrator) to run integrity scanners against the rest of the network to look for signs of other compromised hosts. They confirm that there were no additional impacts.

Next, using a separate Internet connection from a neighboring office, the SA downloads the SANS SNMP tool to ensure that all known vulnerable SNMP software on the organization's network is discovered. The results from the SANS SNMP tool scan match the asset inventory maintained by the organization.

Next, the SA downloads (from the kind neighboring office) the vendor patches applicable for his organization and installs them (again with the help of his trusted friend) on the applicable devices. Next, rule sets on the router and firewall are constructed to perform ingress and egress filtering of ports 161/udp (SNMP),

162/udp (SNMP system management messaging), 199/tcp, udp (SNMP Unix multiplexer), 391/tcp, udp (SynOptics SNMP relay), 705/tcp (AgentX), and 1993/tcp, udp (Cisco SNMP).

Recovery

Since earlier analysis showed that neither the router configuration files nor the IOS software had been modified in any way, the SA realized that no additional modifications were necessary to return the system to a known good state other than what was already done; however, the SA did perform extensive testing on the modified (i.e., patched or filtering rules changed) components to ensure that the patches were working (by using the PROTOS test suite himself) and that they were not causing any unforeseen system problems.

Once these steps were completed, the SA returned the network to its original connected state in accordance with the consensus reached earlier with management. The SA spent the better part of the rest of that day and the next monitoring the system to make sure that the vulnerability fixes held up "under fire." The router and firewall logs showed a number of SNMP scans, but no other adverse effects were seen in relation to these vulnerabilities. The incident handling process had worked!

Lessons Learned

The day after the incident occurred, a meeting was held to discuss the event and reach consensus on lessons learned. The following lessons learned and recommended remediation activities were determined:

- The SANS Top 20 warning against the dangers of SNMP should have been heeded. Another network management scheme should be used. At the very least, the filtering rules added during the incident handling process should have been used to filter SNMP traffic.

- There should have been host- and network-based IDS built into the architecture to alert the security staff of suspicious activity.

- No time should have been wasted in responding to a vulnerability that affected the organization, especially one that was so widely communicated and had such a large population of potential targets.

- The information security manager should take a more active role in ensuring the security posture of the network is adequately maintained through ongoing audits at the very least.

- The organization was fortunate that it had a ready alternate Internet connection source from which to download tools and updates. This source should have been negotiated ahead of time and written into the formal incident handling plan as a reliable source to be used as needed.

References

CERT Coordination Center, URL: www.cert.org

Cisco. Cisco Security Advisory: Malformed SNMP Message-Handling Vulnerabilities. Revision 2.1. http://www.cisco.com/warp/public/707/cisco-malformed-snmp-msgs-pub.shtml

Cohen, Yoram. "Simple Network Management Protocol" URL: http://www.rad.com/networks/1995/snmp/snmp.htm

Deterding, Brent. GCFW Honors Practical. October 11, 2000. URL: http://www.giac.org/practical/Brent_Deterding.doc

Forouzan, Behrouz. TCP/IP Protocol Suite. Boston: McGraw-Hill, 2000.

Foundstone, "SNScan Tool." Version 1.04. URL: http://www.foundstone.com/knowledge/free_tools.html

MITRE Common Vulnerabilities and Exposures Database. URL: http://cve.mitre.org

Oulu University Secure Programming Group. URL: http://www.ee.oulu.fi/research/ouspg/index.html

Red Hat Linux. Errata advisory RHSA-2001:163-23. URL: http://www.redhat.com/support/errata/RHSA-2001-163.html

The SANS Institute. Computer Security Incident Handling: Step-by-Step. Version 2.2. September 2001.

The SANS Institute. "SNMP Tool." URL: http://www.sans.org/snmp/tool.php

The SANS Institute. "The Twenty Most Critical Internet Security Vulnerabilities (Updated): The Experts' Consensus." Version 2.502. January 30, 2002. URL: http://www.sans.org/top20.htm

Schaughnessy, Tom. Cisco: A Beginner's Guide. Berkeley: McGraw-Hill, 2000.

TechnoLogismiki. Hackman Tool available at http://www.technologismiki.com/hackman/index.html.