SNMP – Friend or Foe?

Geoff Huston
March 2002

Ask any network operator what they use for network management and somewhere in the management system, buried far below the on-screen displays, reports and coloured maps, you'll find the Simple Network Management Protocol, or SNMP. Over the past decade SNMP has managed to spread itself far and wide in the networking space, and today SNMP is found not only in the management of internet routers but also in all kinds of switches and transmission elements. SNMP has also spread beyond the traditional boundary of networking management. You'll find SNMP management tools in web servers, personal computers, appliances and even in remote controlled power boards. SNMP even pops up in some 3G wireless handsets. SNMP is not just a network management protocol, but a command and control protocol for an entire world of information technology.

SNMP was developed within the Internet Engineering Task Force (IETF) over a decade ago. Its rapid development and subsequent deployment has often been used as an illustration of the effectiveness of the IETF at that time in producing practical solutions in very accelerated time frames. As its name suggests, SNMP is indeed a simple protocol, and, simple protocols should be able to be implemented quickly and correctly. Or so you'd think. Imagine the level of surprise in early 2002 when a group of researchers working in the area of testing various forms of protocol conformance found that most of the implementations of SNMP in use at the time were found wanting. Many implementations of SNMP were discovered to contain a number of basic errors that in turn had the potential to expose their host systems, and the network they resided in, to quite serious vulnerabilities. This was not simple. This was serious!

Why did it take so long to uncover these vulnerabilities? Why were they there in the first place? To see what happened here its necessary to back up a bit and look at this simple protocol in a little more detail.

The approach used by SNMP is that of a basic model of a management station and a managed device. The management station issues short commands to the managed device, which responds with a command response. As well as command and response, the managed device can also alert the management station to an event that has happened with the device. In this management model, the onus of management complexity lies with the management station, rather than the managed device. There is no persistent connection state between a station and the devices it manages. Each individual command and control transaction is a command from the station to the device and a matching response from the device to the station. A device can be monitored, and even controlled, by any management stations at the same time.

The device management model is also simple. The managed device is expected to continually update an internal set of counters and gauges that describe its current operational state. There may be a counter for the number of packets passed out a particular interface, or a gauge describing the current CPU load, or amount of free memory in the device. Each of these SNMP variables can be polled for their current by the management station at any time (using a "GET" operator in SNMP-speak). The poll is conducted by a simple datagram exchange, where the management station sends a datagram poll to the device specifying a number of SNMP variables, and the managed device responds with a datagram response with the same list of variables, and their current values. Sets of variables are linked in a tree structured name space, and tables of related variables can be retrieved using a "GETNEXT" operator that retrieves the variable that logically follows the specified variable. The inference of all of this is that the management station can construct a remote picture of the operational status of the managed

device by regularly polling a set of counters and gauges that are of interest to the management station. This is the passive monitoring part of network management.

The second part is the command and control component. This is almost identical in form to monitoring the device, except that the management station can pass the device an SNMP datagram with the name of a device control register, and a new value to be placed in that register (a "SET" operation in our SNMP lexicon). Upon receipt of the packet, the device makes the necessary change and responds with the same register name and new value as a confirmation of the action.

The final part of the management model is the device alert function, where the device needs to alert the management station to some operational condition. Again the same format is used, where the device sends an unsolicited message (a "TRAP" in the lingo of SNMP) with the name of the counter or gauge that has exceeded some threshold value, and its current value.

Of course, without some form of access control anyone could play havoc by "managing" anyone else's network. SNMP uses a rudimentary form of access control where a password is used in the header of each SNMP message that passes between the management station and the managed device. These passwords (or "community strings", which completes our little lexicon of SNMP-specific linguistic terms) can determine if a remote management station has read or read/write access to the device.

And that, in a nutshell, is SNMP.

Simple? Absolutely!

SNMP is a simple protocol, and, as a rule of thumb, simple mechanisms can be easily implemented. Why then have some implementations been found to contain serious flaws? All of a sudden SNMP is no longer the network manager's friend, but a source of potential methods of hostile attack on the network and its attached servers. What went wrong to turn this tool into a potential foe?

Interestingly, the efforts of this latest research team were not directed towards finding flaws in the basic protocol exchange used by SNMP, but were directed towards potential flaws in interpreting the internal format of SNMP exchanges, looking closely at the encoding of SNMP's variable fields that use a structured encoding of a triplet of a type field, a length field and a value. This encoding is perhaps one of the stranger aspect of SNMP. Prior to SNMP there was a strong tendency to use very simple identification of elements within various Internet-related protocols, relying often on plain text within the protocol exchange to describe actions and values within the protocol. For example, mail exchange via the SMTP protocol is a conversation of quite readable text fields. SNMP chose a different path, and instead uses a formal specification language, Abstract Syntax Notation 1 (ASN.1), to describe data fields within the protocol. ASN.1 Basic Encoding Rules (BER) are used to encode the variables within the protocol itself. This was a novel approach at the time, in so far as the use of formal languages to describe protocols and their data objects was more often seen in an academic computer science curriculum than in the wild of real network protocols. Even today few protocols have repeated this particular step.

And perhaps its a good thing.

If this use of formal language to specify the protocol elements was such a novel approach at the time, then its reasonable to ask why did the designers of SNMP choose use ASN.1 and the corresponding encoding rules for SNMP? To use a quote from one of SNMP's designers, Marshall Rose, in "The Simple Book", in an admitted non-technical commentary, Marshall notes:

> *"The official reason for using ASN.1 is to ease the eventual transition to OSI-based network management protocols. The actual reason is that the Internet research*

community got caught napping on this one, having never spent much time dealing with application-layer structuring. Fortunately, ASN.1 is destined, for better or worse, to become the network programming language of the 90s, just as the C programming language is largely seen as having been the systems programming language of the 80s. So, the choice of ASN.1 is a good one."

Fortunately, for the world of programming, this foreboding prediction was never to eventuate in the 90s, and the pain of ASN.1 is largely confined to the world of SNMP data objects.

And it has proved to be a pain. Despite all this faith in formal languages and the concept that formal languages lead to precise and deterministic formal language parsers, that in turn lead to precise and deterministic protocol implementations, the reality exposes a bad case of misplaced faith. Noone had ever stress-tested SNMP implementations before simple because there was a strong case of belief that anything that was so carefully specified must be correctly implemented. The SNMP test suite in question generated a series of cases where the encoding rules were deliberately broken. The suite used various illegal code points, incorrect length fields and nonsense value fields to test whether an SNMP implementation can correctly detect and discard poorly formatted SNMP messages. And here's where various SNMP implementations were found wanting. A number of implementations attempted to process the poorly encoded packet in any case, leading to results which, in the best case simply confused the SNMP process, and in the worst case corrupted memory so badly that the device crashed or allowed foreign capture of the device. Whoops.

It should be obvious that within any protocol exchange one party cannot trust the other to send correctly formatted data in a well behaved pattern. Various forms of corruption of packet contents can occur, oversized giant packets do happen, and packets can be sent at flooding rates. These approaches, and more, have been used as forms of hostile attack in the world of network protocols for decades, and SNMP is not excluded from the cut and thrust of network level attacks. Implementors of protocols make limiting assumptions about what kinds of input they receive at their peril.

In the case of SNMP it appears that some implementations assumed that it was just not possible to see as input incorrectly formatted ASN.1 encodings. The implementation was not configured to reject encodings when the input packet was incorrectly formatted. Even when the internal fields contained nonsense values, the implementation persisted in decoding the fields and acting accordingly. In the best case the SNMP implementation worked on the principle of 'garbage in, garbage out", and simply returned nonsense responses. The the worst case these badly formatted packets caused the implementation to corrupt its memory maps. The researchers' report noted that this lead to quite dire outcomes, with some implementations causing the managed device to enter a loop of continuous rebooting, and others allowing the device to then run injected code, thereby allowing the device to be hijacked by the attacker. All this would be bad enough, but it was also noted that in some cases the SNMP implementation failed even when an incorrect community string was included.

Given the prevalence of vulnerable implementations of SNMP, and the capability of disrupting the management system without knowledge of any passwords, he upshot of all this protocol conformance testing was message of some considerable concern to network operators. The corrective response from the vendor and operations community has evidently been quite effective, and there has been no evidence that any exploitation of this vulnerability has caused undue disruption so far. But it still leaves the lingering suspicion that the life of a network management protocol is no longer as simple as it once was.

The major architectural question that this exposes is the vulnerability of in-band management and control systems when coupled with a datagram-based query/response model. While in-band control systems are often an efficient answer to management requirements, the mixing of

control flows and data flows within a common plane in the network exposes the capability of injected data attempting to masquerade as control information. The lack of a connection state prohibits the management station and the managed device from establishing an authenticated trust relationship based on key exchange and encryption of the dataflow. The SNMP response is for the management elements to use a combination of passwords and address-based access filters to attempt to filter out such illegitimate management traffic. But it really is just not good enough. Many forms of hostile attack are based on forging the address fields in a data packet, and, as this exercise exposed, in some cases it was not even necessary to have knowledge of the access password to disrupt the operation of the device.

Maybe SNMP is just too simple to be the network manager's friend. It is becoming obvious that higher levels of network resilience can be achieved by moving away from a common data and command and control plane into some form of a segmented architecture. Also there is some resilience in moving to a protocol that used a shared state between the manager and the managed, allowing the two network elements to thereby support a private trust relationship. But all this is getting into a topic that remains still in the area of active research, while the lessons to be learned from this incident are perhaps quite simple:

The mantra of the implementor for Internet-based applications is the oft-quoted phrase from Jon Postel, who, in the Internet Protocol specification of 1981, cautions implementors to "*be liberal in what you accept*", But this is evidently not enough. The other adage of being "*very devious and broadminded of what you will need to reject*" also applies within the Internet!

For those interested in reading the research report on protocol conformance tests of SNMP, the report can be found at: http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/