PROTECTING SNMP THROUGH MARKETNET

Table of Contents

Marcela Jackowski

I. INTRODUCTION

As dependency on information technology becomes more critical so does the need for network computer security. Because of the distributed nature of networks, large-scale information systems are highly vulnerable to negative elements such as intruders and attackers. The types of attack on a system can be diverse and from different sources. Some of the factors contributing to creating an insecure system are the relentless pace of technology, the need for information processing, and the heterogeneity of hardware and software. In addition to these insecurities, the growth and success of e-commerce make networks a desirable target for intruders to steal credit card numbers, bank account balances, and other valuable information. This paper looks at two different security technologies, SNMP v3 and MarketNet, their architectures and how they have been developed to protect network resources and services, such as, internet applications, devices, and other services, against attacks.

II. EVOLUTION OF SNMP

SNMP, Simple Network Management Protocol, is a network management tool that aids in the monitoring and controlling of network devices. SNMP remotely controls the activities of network devices through polling, setting terminal values, and detecting network events. SNMP's main goal is to keep the performance of network devices running as productive and efficient as possible. SNMP was originally created out of necessity as a makeshift tool to manage networks and the inherent complexities of such. In its beginning, SNMP's goal was to provide a temporary answer until a more robust network management tool could be designed, however, because it did address short-term network management needs it became widely used.

The key feature to SNMP's popularity was its availability and, at the time, it was considered simple to use and easy to install. SNMP works by exchanging network information through protocol data units (PDU's) supported by TCP/IP. SNMP v1 was the first attempt to create a standarized software that could meet the demands of managing network devices. SNMP was designed in the mid-80s and the initial purpose was never to be a self-protective software nor a fool-proof security system but rather a "simple" network monitoring system. The enormous expansion of networks in the late 80's tested SNMP v1's ability to manage network systems and deemed it deficient. The

testing of SNMP proved it to have two major flaws. The first deficiency was system security; SNMP v1 does not authenticate the source of a management message nor does it prevent eavesdropping. The second deficiency was the inability to support distributed network management; SNMP v1 does not have the ability to support the management system to share information about other management systems' devices or the networks they manage.

The framework of SNMP v1 is:

1. a protocol (a set of network communication specifications).
2. a management system (the manager).
3. an agent.
4. a management information base (MIB).

Although all four components of SNMP are important, the real essence of the system lies in the SNMP protocol itself. This protocol provides a central mechanism that controls operations and is designed to accommodate the exchange of management information between manager and agents. The PDU, the protocol data unit, is the elementary unit of communication in this protocol. It is a packet or message that contains data that needs to be transmitted to the agent plus control information. SNMP v1 has 3 PDUs available for the management station to communicate with its agents:

SetRequest, sets the value of an object instance within an agent.
GetRequest, requests the value of an object instance from an agent.
GetNextRequest, requests the value of the next object instance from an agent.

The above functions are initiated by the management station. The Set PDU function provides the management station with writing privileges to the MIB, for example, to enable or disable the state of a link the Set PDU is used. To retrieve a value from a MIB, the Get PDU function is used. The Get PDU has read-only privileges and can retrieve the value of an object.

Agents have the following 2 PDUs available to reply to managers' requests or to inform about network events:

GetResponse, this is an acknowledgement to any of the three functions initiated by the manager.

Trap, this PDU is used by the agent to take initiative and indicate to the manager that an event in the network was detected.

The overall architecture of an SNMP application is based on the client/server model. The SNMP management component, the management station, corresponds to the client application. It communicates and generates requests to the agents. The SNMP manager is responsible for maintaining and updating network resources and its configurations, for example, setting values of variables that trigger alarms. Among other tasks, it polls the agents for information stored in the MIB's. Through these queries to the agents, the manager is able to monitor and control network conditions. The agent acts as the server by replying to the requests made by the manager. The function of the agent is to respond to requests and actions set by the management station, collect and store network information in its MIB, and communicate network information to the management station through polling and event reporting.

The management information base component, the MIB, is a database that stores information about the entities being managed. It defines variables for which data is maintained and updated. The organizational structure of the MIB is a hierarchical tree. This hierarchical tree structure represents the different devices, components, and status of the network. The top level of the tree maintains general information about the network and resources to be managed. The leaves of the tree represent the MIB objects. Each branch of the tree has a unique name and numeric identifier. The tool used by the management station to access MIB information is the PDU.

II.1 Enhancements made to SNMP V-2

As networks rapidly proliferated in the 1990's, the urgency to automate network monitoring and controlling became acute and it was confirmed that SNMP v1 could not sustain the new requirements. The two major flaws; its lack of support for a distributed network management and its inability to provide a secure network system had to be corrected.

To address these two problems a new version of SNMP was implemented, SNMP v2. V2 offers functional enhancements and features that improve efficiency over v1. First, communication from manager-to-manager was implemented in order to support multiple or distributed managers. This feature allows management stations to communicate with each other and provides the ability to monitor networks instead of monitoring only the nodes of a network. Second, it improved the efficiency in the retrieval of data by creating two new PDUs:

GetBulkRequest PDU, retrieves large blocks of data

InformRequest PDU, allows managers to send trap type information to other managers and receive a response

Third, it modified the response of some of the PDUs. For example, PDU GetRequest in version 1 was implemented to either provide values for all the variables or none. GetRequest in v-2 was modified to supply a partial list of the values requested instead of no data at all.

While several attempts were made to implement a secure system through various version enhancements of v-2, such as SNMPv2u and SNMPv2*, the results were unacceptable due to its high level of complexity, its incompatibility with v1, and the lack of support from the Internet Engineering Task Force [1]. The continued growth of networks, the absence of a substitute for SNMP, and the less than desirable results of the second version led the SNMP working group to consider a third version.

III. SECURITY ENHANCEMENTS TO SNMP

SNMP's third version is not a redesign of v1 and v2. It is v1 and v2 plus security enhancements and its primary goal is to attempt to correct the critical issue of network security that eluded versions 1 and version 2. The architecture of v3 consists of a modular framework where it incorporates system security as an add-on. The top level architectural design for v3 is composed of two major modules that service each other in the processing of incoming and outgoing messages:

1. **SNMP Engine**
2. **SNMP Applications**

An SNMP engine plus and SNMP application can be either a management station or an agent. The difference is determined on whether the SNMP engine contains the Access Control module which is used solely by the agents. V3 architecture defines new terms of entrustment within a network system which we define now: SNMP entity, context engine, principal, authoritative engine, non-authoritative engine, SNMP engine and SNMP applications.

III.1 SNMP ENTITY

The architecture of an *SNMP Entity* consists of two modules, see Fig. 1. The first module is the SNMP Engine which has four components: *Dispatcher, Message Processing Subsystem, Security Subsystem,* and *Access Control Subsystem.* The second module is the SNMP Applications, composed of : *Command Generator, Command Responder, Notification Originator, Notification Receiver, and Proxy Forwarder.*

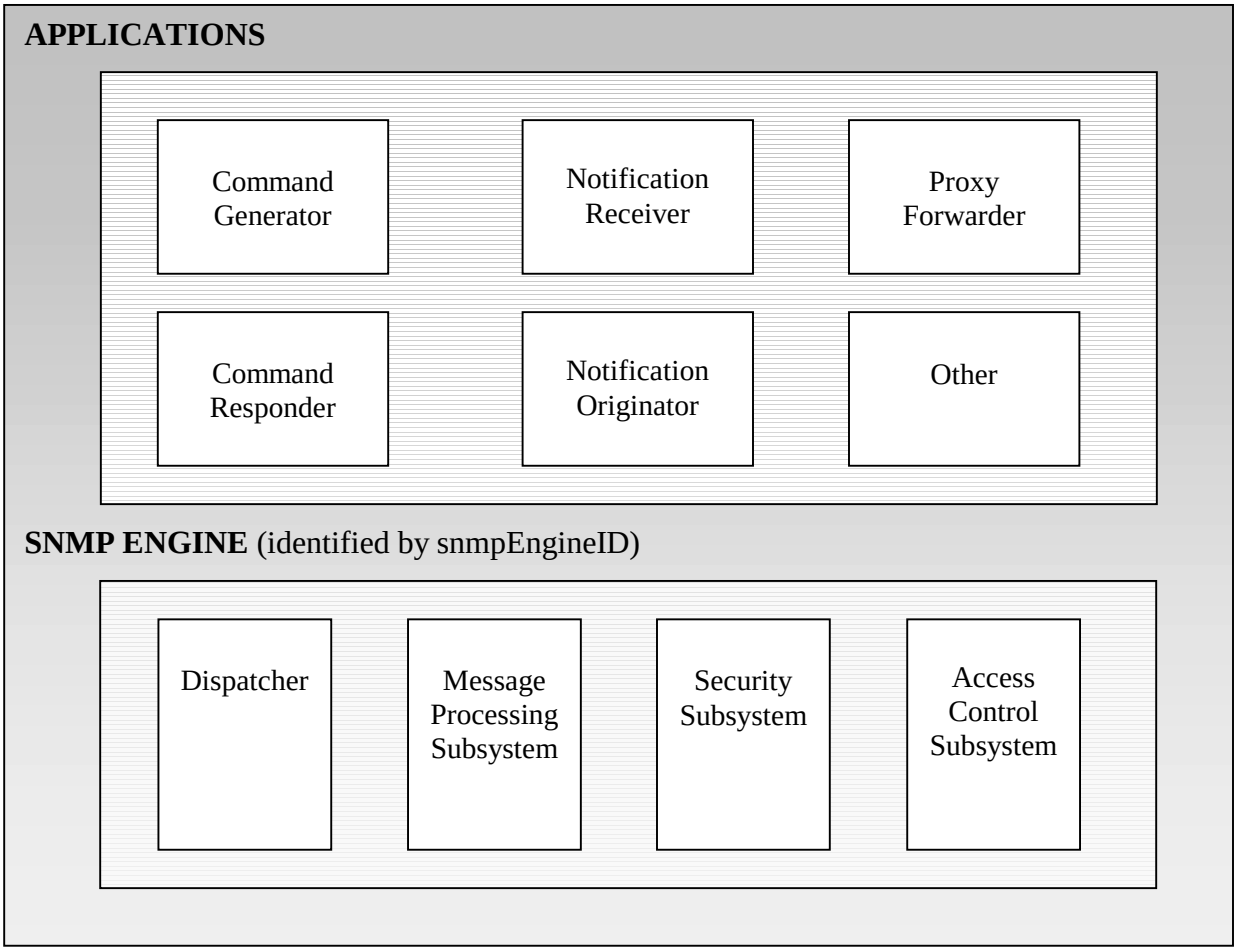
An ***SNMP Entity*** may be an agent, a manager, or a node working as both agent and manager. As an access control security measure each *SNMP Entity* has a single *SNMP Engine.* The SNMP Engine is distinguished by a unique and unumbiguous identifier, *snmpEngineID.* This identifier matches the identifier of the SNMP Entity that an SNMP Engine belongs to. This unique identifier maps the SNMP *Engine* to only service those entities that have a corresponding matching identifier. Requests made by entities whose identifier do not match are discarded by the SNMP *Engine,* thus providing protection from potential intruders.

SNMP entities, such as an agent, have a ***Context Engine*** or context manager that is responsible for managing one or more contexts. The context engine has a unique identifier, *contextEngineID.* The *contextEngineID,* uniquely identifies an SNMP entity that wishes to transact an instance of a context with a particular *contextName.* Each managed context has a unique *contextName* within the entity. The terms, context engine, context, and SNMP engine work together as a security concept in access control. To establish access control there is a one-to-one correspondence between context engine and the SNMP engine at an entity, the *contextEngineID* is identical in value to the *snmpEngineID.*

The term ***Principal*** refers to the entity requesting access or services. In general, a principal operates from a management station and issues SNMP commands to an agent. Different principals have

different levels of responsibility and security. Depending on the access privileges of the principal varied security features will be implemented and coordinated between the principal and the agent.

Figure 1: Architecture of an SNMP Entity



The term ***Authoritative Engine and Non-Authoritative Engines*** is another security related concept defined for v3. To control message sending and receiving one of the two entities is defined as the authoritative SNMP engine, according to the following rules:

- when an SNMP message contains a request which expects a response (for example, a Get, GetNext, GetBulk, Set or Inform PDU), then the receiver of such messages is authoritative.

- when an SNMP message contains a request which does not expect a response (for example, an SNMPv2 Trap, response, or Report PDU), then the sender of such a message is authoritative.

The concept of authoritative status provides two security controlling features. First, the authoritative engine controls the timeliness of a message. It prevents delays of messages and replays by setting a reasonable time frame for the message to be delivered. The time frame cannot exceed the time limit because this would indicate a high risk of malicious delays. Nor can the time frame be too small that a genuine message could be discarded as illegitimate. And, two, it provides key localization control. An authoritative engine holds a unique key for every authorized user. A localized key is defined as a secret key shared between a user and one authoritative SNMP engine.

III.2 SNMP ENGINE

The ***SNMP Engine*** module is responsible for coordinating the sending and receiving of messages, authenticating and encrypting or deciphering messages, depending on whether the messages are incoming or outgoing, and controlling access to MIB objects. This modular framework addresses three important security issues: privacy, authentication, and access control. Some of SNMPv3s security features are directly inherited from SNMPv2u and SNMPv2*, while other security features have been specifically defined for SNMPv3.

SNMP ENGINE COMPONENTS

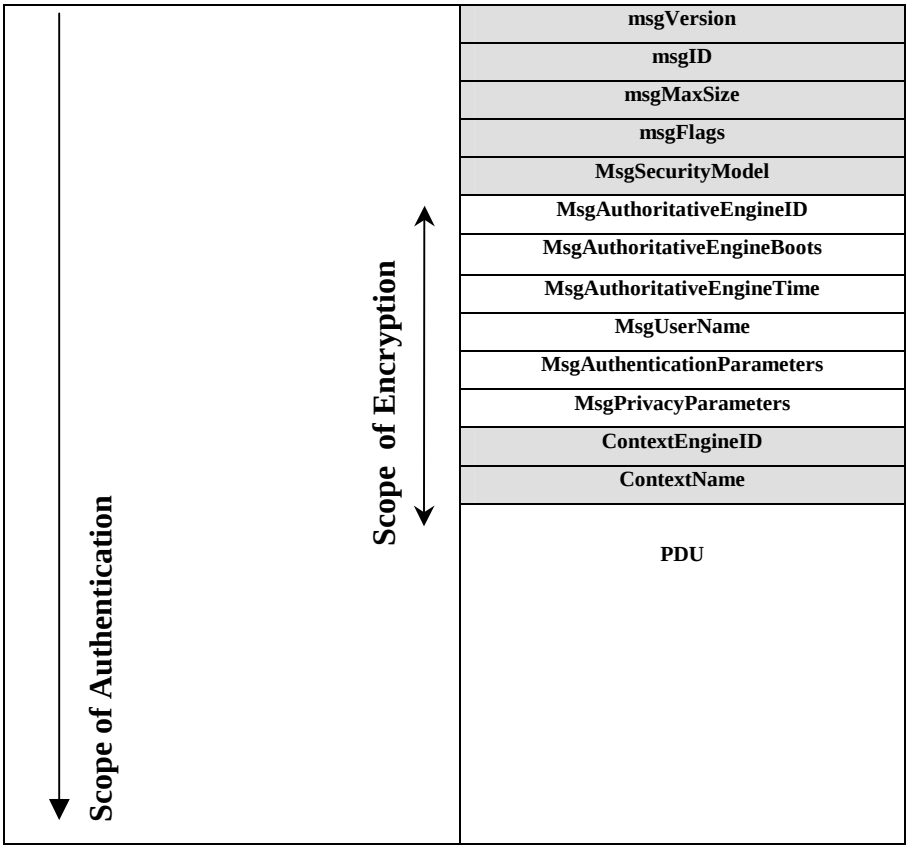
DISPATCHER

The *Dispatcher* works as a message reception/delivery manager. For outgoing messages, the *Dispatcher* determines the PDU version, i.e., v1, v2, or v3, and accordingly directs the PDU on to the appropriate application or message processing module in the *Message Processing Subsystem.* Once the *Dispatcher* receives the outgoing PDU back from the Message Processing Subsystem, it transmits the message over the transport layer. For incoming messages, the *Dispatcher* will accept messages from the transport layer, determine the version type of the message and route it to the appropriate *Message Processing Subsystem.* Once the PDU is received from the *Message Processing Subsystem* it delivers it to the appropriate application.

MESSAGE PROCESSING SUBSYSTEM

The ***Message Processing Subsystem*** accepts messages from the *Dispatcher* and processes the message header, (see Fig.1).

Figure 2: SNMP Message Header Format



The *message processing subsystem* is responsible for processing message headers of incoming and outgoing PDUs. For incoming messages, this module analyzes the following five parameters on the messages' header plus *ContextEngineID* and *contextName*:

msgVersion - defines SNMP version.

msgID - unique identifier used between two SNMP entities.

msgMaxSize - contains maximum size of message.

msgFlags - defines the security level: provide encryption, authentication and whether a Report PDU is requested.

msgSecurityModel - determines the security model used by the sender so that the receiving entity uses the same.

ContextEngineID and contextName provide information that allow the PDU to be unambiguously identified. For incoming messages, the *Message Processing Subsystem* strips the information from the first five fields of the Message Header, See Fig. 1, for processing. It passes this information to the user-based security model (USM) which authenticates the validity of the security related information on the message header. For outgoing messages, the *Message Processing Subsystem* is responsible for creating the message header for the PDU being processed and invokes the user-based security model (USM) with the necessary information to provide authentication and privacy to the outgoing message. Once the *Message Processing Subsystem* finishes its tasks, whether it is an outgoing or incoming message, the message is sent back to the Dispatcher.

SECURITY SUBSYSTEM

The *User-Based Security Model (USM)* is one of two SNMP v3 add-on security module. Its chief function is to provide security by scrutinizing the validity and authenticity of incoming and outgoing messages. The USM model corrects the two critical security deficiencies found in SNMP v1 and v2; i.e., the inability to authenticate the source of a management message and the prevention of eavesdropping. This module is also responsible for time synchronization to prevent delays and replay attacks. It includes a key management capacity; key localization and key update. to support its cryptographic functions. A discovery security mechanism is also implemented in this module. The discovery process provides information about other SNMP engines in order to communicate with them. The *User-Based Security Model* operates at the message header level of the PDU and communicates with the Message Processing Subsystem in the handling of outgoing and incoming message.

The *User-Based Security Model* is responsible for either creating the following fields in the message header for outgoing messages or it processes the following fields for incoming messages:

msgAuthoritativeEngineID: contains the snmpEngineID of the authoritative SNMP engine involved in the exchange of this message.

msgAuthoritativeEngineBoots: contains the value of snmpEngineBoots for the authoritative SNMP engine involved in the exchange of this messages.

***msgAuthoritativeEngineTime*:** contains the value of snmpEngineTime for the authoritative SNMP engine involved in the exchange of this messages.

***msgUserName*:** the user on whose behalf the message if being exchanged.

***msgAuthenticationParameters*:** value to determine if authentication is requested for this message.

***msgPrivacyParameters*:** value to determine if privacy is requested for this message.

The USM prepares messages for transmission in the following sequence, if encryption is requested by the entity then the scoped PDU is encrypted and placed on the message header, see field in Fig. 1, and the value of field *msgPrivacyParameters* is flagged to the appropriate value. If authentication is requested the entire message, including the scoped PDU is input to HMAC, and the resulting authentication code is placed in *msgAuthenticationParameters*.

The *User-Based Security Model* implements two cryptographic functions to protect against false identity and privacy attacks: encryption and authentication. Encryption and authentication are at the heart of the SNMP v3 security system. The concept of encryption and authentication share similar characteristics, both the sender and receiver must share a secret key. These cryptographic functions have two inputs: the actual text plus the secret key, and they both use an encryption algorithm. The secret key for both of these functions must be preconfigured by a network manager. The values of privKey and authKey are not available via SNMP.

The authentication function used in SNMP is the Hash Message Authentication Code, HMAC. HMAC, uses the MD5 (message digest algorithm) as the hash function. This hash function accepts a variable-length message M as input and produces a fixed size hash code H(M) as output [4]. This algorithm attempts to assure system security by providing access to only authorized entities and by protecting the message from being modified during transmission. Authentication works by having the sender and receiver share a secret key, *authKey*. This secret key is set up manually by the system manager during configuration. Using the secret key, the sender generates a small block of

data named message authentication code. Included in the message processed by the sender is the following identification information: identity of the principal and engine, the time of transmission, and the secret key to decipher the code. The sender includes the message authentication code plus the key in field *msgAuthenticationParameters* in the message header. The receiver must have the same secret key to be able to compute the message authentication code. If the match between the received code and calculated code is successful, then the receiver is guaranteed that the message was not altered and that the message's identity is authentic. In fact, it guarantees that the message comes from the principal identified in the message. Authentication prevents intruders from altering in transit messages and the masquerading of authorized entities.

Another important security element which enhances the security of a system and that is used along with authentication is the timeliness of a message. By encoding the time of transmission in the authentication message, the receiving entity can assess the timeliness of the message and verify that the message was not delayed or replayed. SNMP engines, receiving and sending entities, work in a way such that their clocks have been synchronized so that both entities can estimate the time needed for a message to reach its destination. The two parameters dealing with the timeliness issue in the message header are, msgAuthoritativeEngineBoots, which contains the value of the number of times the engine has initialized or reinitialized itself since its initial configuration. And, msgAuthoritativeEngineTime, which contains the time value for the authoritative SNMP engine involved in the exchange of this messages.

Encryption is the second cryptographic function used in SNMP v3. This function protects information against eavesdropping. Encryption protects against modification of data and, like authentication, it assures that the data will be made available only to authorized parties. The *User-Based Security Model* uses the cryptographic function of cipher block chaining, CBC, from the Data Encryption Standard model. Like authentication, CBC, works by having the sender and receiver share a secret key, *privKey.* And, like authentication, encryption has two elements: the messages to be encrypted and the key. Using the same key, the receiver deciphers the message using the DES algorithm. If encryption is desired the value of MsgPrivacyParameters is set. This field holds the value needed to generate the initial value, IV, in the DES CBC algorithm. The sending and receiving parties' secret key is set up manually by a manager during system configuration

View-Based Access Control Model (VACM)

The *View-Based Access Control Model (VACM)* is the second security capability defined for SNMPv3. The *View-Based Access Control Model (VACM)* is part of the agents' SNMP engine and is one of four components residing in the agents' SNMP engine. This module is responsible for controlling authorized access to managed contexts and operates at the level of the SNMP PDU. VACM determines the access control privileges that a requesting principal has, such as, read, write, or notify. Based on the principal's security model and security level, VACM grants or declines access to managed contexts. The access control policy used by an agent to determine the privileges of a manager are preconfigured in the network system. A hardwired table contains the access privileges of the different authorized managers been serviced by the agent.

The View-Base Access Control model defined for SNMPv3 implements two security measures against unauthorized users: first, it enforces access control policies to remote managers requesting access to MIBs for the reading and setting of management objects, and, second, the VACM model determines which MIB access privileges should be available to managers.

There are five concepts working together to exercise access control in the VACM, these are: ***groups***, ***security level, contexts, MIB views,*** *and* ***access policy***:

> ***Groups***, is the way users are organized in order to obtain access control to managed objects.
> ***Security level***, this concept determines the level of security sensitivity a message should have. Depending on the sensitivity level of a message an agent may require that the principal request be done with privacy or authentication service, or both.
> ***Contexts,*** refers to the way in which object instances are organized so that different principals have access to different managed objects. For example, a particular principal has access to particular managed objects in the MIB.
> ***MIB Views***, is the way managed objects are grouped so as to be able to provide different levels of services to match the security level of the manager princiapl as well as to provide security by fending off unauthorized users.

Access policy allows an agent's SNMP engine to determine different levels of access rights to the principal requesting access to a MIB. Depending on the manager or principal, the type of access requested, the MIB context, and security different access policies apply.

The VACM security system works with a corresponding VACM MIB that supports access control. Just as the VACM maintains preconfigured lists of which entities have access control to managed objects and the types of security levels, the VACM MIB also maintains a preconfigured list of contexts names (vacmContextTable list) and other key elements, such as, groups, access rights, and MIB views.

III.3 SNMP APPLICATIONS

The applications associated with an SNMP engine are determined by the type of the SNMP Engine, whether it is an agent or a manager. The three following applications reside in an SNMP agent and are responsible for generating and receiving SNMP PDUs; Command Responder Application, Notification Originator Application, and Proxy Forwarder Application. These applications work with the services provided by the Dispatcher, Message Processing Subsystem, Security Subsystem, and the Access Control Subsystem. The applications implemented for an SNMP management engine are; Command Generator, Notification Application, and Notification Receiver Application. These applications generate and receive PDUs and are responsible for monitoring and manipulating management data at remote agents. The communication mechanism used between the SNMP engine and its applications is done through primitives and parameters. "A primitive specifies the function to be performed, and the parameters are used to pass data and control information."[2]

COMMAND GENERATOR

The *Command Generator* monitors and manipulates management data at remote agents. It supports SNMP PDUs from v1 and v2. It generates Get, GetNext, GetBulk and Set request PDUs and processes the response provided by these requests. The *Command Generator* uses the sendPdu primitive to generate its PDUs. The sendPdu primitive contains information about the PDUs destination, security parameters and the actual PDU. The parameters used in sendPdu() are Taddress

which includes transportDomain, and transportAddress, Cname which contains the values for contextEngineID and contextName.

Once the SNMP engine receives the sendPdu() primitive from the *Command Generator* then the processing of the message begins with the *Dispatcher*. The *Dispatcher* takes two actions upon receipt of a request from the Command Generator: first, it sends a prepareOutgoingMessage() primitive with parameters received from the Command Generator plus PduHandle to the *Message Processing Subsystem*. Second, it assigns a unique sendPduHandle to this PDU.

The *Message Processing Subsystem* receives the primitive prepareOutgoingMessage() with the above parameters and takes the following action:

(1) it assigns a unique msgID to the message requested.
(2) it prepares primitive generateRequestMessage() to be sent to the *Security Subsystem* with the contextEngineID, contextName, PDU, msgGlobalData from the message header plus msgID, and sets msgFlags according to the securityLevel received from the *Command Generator*.

The *Security Subsystem* receives the request primitive from the *Message Processing Subsystem* and prepares the message according to the required specifications. The task of the USM is to determine the validity of the security parameters, such as, check if the PDUs destination is valid, if the securityLevel requested is supported for this user. If authentication is requested, the entire message plus the secret key is encrypted using HMAC. If privacy is requested, the scopedPdu plus the secret key is encrypted using CBC-DES. The end values of authentication and encryption will respectively fill in the fields msgAuthenticationParameters and msgPrivacyParameters in the message header. USM returns the message with the appropriate security validations to the *Message Processing Subsystem* that in turn returns to the *Dispatcher*.

The Dispatcher ends the processing of the generated outgoing request message by passing the message down to the transport layer for transmission and returns the sendPduHandle to the Command Generator. The value sendPduHandle is used as a unique identifier for this PDU so as to keep track if later there is a matching response to this particular PDU.

COMMAND RESPONDER

This application resides at the agent's module. The *Command Responder* application services incoming request messages from other SNMP entities. It accepts incoming SNMP PDUs requesting to either fetch the value of a managed object or set the value of a managed object plus provide the requesting entity with a response message. The Dispatcher coordinates the handling of the incoming messages for the command responder application. Before passing the incoming request to the command responder, the Dispatcher determines if this message has obtained clearance from the various modules in the SNMP engine. During this process, if any of the parameters in the incoming message are invalid then the message is discarded. Otherwise, if the incoming message is successfully processed at the Message Processing subsystem and the USM then the Dispatcher passes the PDU to the appropriate application. The Dispatcher determines the correct application for this incoming message by examining which application has registered for this contextEngineID and pduType.

Once the message has been passed to the command responder by the Dispatcher, the command responder proceeds to check if the message is actually authorized to access the information it is seeking by getting clearance from the access control subsystem. If the response from the access control subsystem is successful then the application proceeds to process the request PDU and prepares a Response PDU. The response for the incoming request message is prepared by the Command Responder, it sends a returnResponsePDU primitive to the Dispatcher.

The ***Notification originator*** application works as a monitoring device. It generates Trap and Inform PDUs to specified managers regarding system events. This application uses the same message processing pattern and the same primitives as the Command Responder Application, that is, processPDU and sendPdu for Inform PDUs and sendPdu only for Trap PDUs,.

The ***Notification Receiver*** application receives Trap and Inform PDUs. This application works in a similar way to the Command Responder Application in the processing of incoming PDUs. The Notification Receiver must first register to receive Trap and Inform PDUs.

The ***Proxy Forwarder*** application forwards SNMP messages.

IV. MARKETNET

MarketNet is a dynamic network security technology based on economic models. The goal of this new technology is to guarantee protection from attacks and operate efficiently over very large distributed networks. In contrast to traditional network security systems MarketNet dynamically manages its resources and regulates its risks. This new technology does not rely on traditional protection mechanisms such as, binary access control, ad-hoc programming, or customized software to each component, which have proven to be vulnerable to attacks. The incentives attackers have to crack a system are ample. Undetected attacks can strike at any time, from anywhere, in a variety of forms. For example, financial institutions and banks, are likely to be probed and attacked in attempts to commit fraud. An attacker can take down the network of a financial institution for a certain amount of time, resulting in massive financial losses. Another incentive for attacking a network is industrial espionage. In this case, the attacker will steal valuable data and key technology developments either to be sold or improve the competitors knowledge and standing in the market. In either case, the targeted entity will incur severe financial losses. This section is based on material in [1], [2], [3].

IV.1 Traditional Security Mechanisms

Traditional protection technologies do not guarantee the necessary security against attackers and the challenge of protecting a large-scale system is a moving target. First, traditional protection technologies lack a systematic and continuous monitoring access control to resources. Typical protection technologies, such as, binary access control, ad-hoc programming, or customized software for each component, fail to detect dangerous intrusion attempts and fail to provide accountability for an attack. Vulnerabilities such as these provide vast opportunities for attackers. Security technologies, such as, binary access control, fail to guarantee network security in several ways. It grants access control to anyone who has the right password. Regardless of how the password was obtained, whether legitimately or not, access is granted. Binary access cannot determine the validity of the user or its ulterior motives. Once the user accesses the system, the ill intentioned intruder is free to roam around and attack the system. After having granted access to an unauthorized user, binary access does not support either preventive control, to avoid the occurrence of unwanted events, nor detective control attempting to identify or isolate the event. Furthermore, binary technology does

not implement the sharing of information of an intrusion or anomalous access with other components. Having the intruder conquered the system there is no protection binary access control can provide to deter the intruder or minimize any further damage. Finally, system networks are not only vulnerable to external attacks but also internal attacks. Many of the real damaging incidents of computer crimes are carried out by insiders. If the attacker is an insider, not only does the perpetrator have a valid password but is also able to manipulate its own audit trail. Again, binary access offers no protection to an internal attack.

A second vulnerability identified on traditional protection technologies is the lack of a uniform security mechanism. The lack of a uniform security mechanism creates the persistent need for specialized security mechanisms. The fast proliferation of systems and constant changes in technology all contribute to creating an heterogeneous network where different components, software and hardware from different vendors, have different security needs. This heterogeneity contributes to creating new security gaps within a system. Specialized security mechanisms are highly costly since they are tailor made for different components and on a as needed basis. The time needed to solve a security problem may deliver results long after the attack has occurred. And, the money needed to pay expert help can result in financial losses for the target network. The lack of uniformity to defend a system requires expert manual labor to monitor and correlate access anomalies in order to detect an attack. The major liability created by an heterogeneous systems, is that by the time security controls are in place the response is untimely and ineffective. Most likely, the attacker is left unaccountable for damages and has evaded identification detection.

A third and severe security vulnerability found in traditional security technology in networked systems is the exploitation of trust among components. In other words, components offer services to each other based on trust. Many large networks are commonly attacked through the exploitation of trust between hosts. An attacker will look for trusted network components to break into, such as a server that is regarded as secure. For example, before setting out to attack a network from the internet, a typical attacker will perform some preliminary probes of the networks external hosts present on the internet and try to gain access. The attacker will build lists of external and internal hosts. Through some analysis from his probing information the attacker will start to identify trust between hosts and attempt to gain access. Once the intruder identifies a trusted host he/she will use it as a point from which to launch an attack on the network. From this point, the attacker will usually

gain access to other hosts. Once the system is conquered, the attacker can set up a clean-up process so as to have undetected access to the host later. Furthermore, the attacker can ensure that his or her presence has not been logged in any way and keep abusing the system undetected and indefinitely. In this case, even if the attacker is detected it will most likely evade accountability for its crime since it will almost be impossible to follow its trail and track down.

Marketnet is a novel approach to protecting large-scale networks. It implements new security mechanisms desisting the vulnerabilities of traditional security technologies. MarketNet's focus is on shifting power from attackers to defenders. This new technology provides a dynamic security system which tries to anticipate and eliminate security threats by continuously monitoring its clients thus ensuring that attackers do not get a chance to compromise security in the first place. Marketnet provides an efficient and effective security mechanism where distrustful entities are serviced in a uniformed and controlled way regardless of their ulterior motives.

IV.2 Overview of MarketNet's Security Mechanism

In this section we will discuss how MarketNet's novel technology contributes to providing a proficient security mechanism to large-scale networks. Unlike traditional security technologies MarketNet is able to:

1. Service distrustful hosts.
2. Service inside and outside clients.
3. Empower defenders by allowing them to control their exposure to attacks.
4. Identify and hold attackers accountable for their crimes.
5. Provide a systematic and continuous monitoring system through its IDM, Intrusion Detection Monitor.
6. Be readily applied to heterogeneous systems with low-overhead and transparency.

Electronic currency is the key concept used in MarketNet to provide access control and use of its resources while providing security to the network system. Currency provides a uniform instrumentation to measure resource access. This electronic currency contains valuable information that shifts power from attackers to defenders. MarketNet organizes resources and services in

currency domains where each domain controls its currency. Currency domains require that clients pay an access price in the denomination of the target resource. Currency domains are the central units of protection and maintain full accountability on the entities to which they have disbursed currency, and are liable for providing trading access rights. Currency domains provide global security to networks by having each domain generate, control, and monitor its own currency. This systematic detective control provides the necessary information to trace the origination of an attack, to assess the damage of an attack, and as a means to establish full accountability in the use of resources. In addition, a currency domain will activate its intrusion policies and inform all other domains if a system violation or potential attack is detected.

Currency domains are responsible for:

- Issuing its own currency.
- Directing its resources to use the domain's currency for access control.
- Allocating budget to internal and external users.
- Controlling the pricing of resources.
- Activating its internal intrusion policies.

The protection mechanism created by electronic currency is that a domain can defend itself and other domains by eliminating new budgets or voiding currencies, effectively isolating the attack source. Isolating the source of an attack reveals the identity of the intruder through the unique currency identifier and enables to determine the source domain and owners of the currency used in the attack. The target domain will report the identity of the attacker to other domains, thus preventing further abuse.

The currency used in MarketNet contains key information that uniquely identifies the issuer, the current owner, usage purpose, and is not forgeable. The use of this uniquely identified currency plus the fields listed below establish full accountability in the use of resources by tracing access to resources back to the holder of the currency. MarketNet's currency used by a domain has a unique identifier so that a client can prove its identity to the server and vice versa. This unique identifier provides a fast tracking mechanism to its origin in case of malicious attacks.

The following fields provide proof of the currency:

- Currency domain's exchange bank
- Unique ID
- Amount
- Validity
- New owner's exchange bank
- Purpose
- Provider
- Timestamp

To access a resource, MarketNet requires that an access price be paid in the currency of the target domain. MarketNet defines and enforces pricing policy through currency domains. Currency domains have full control of service prices, resource prices, and full control of the allocation of budgets to clients. The currency dissipation policy implemented by domains is a very powerful tool in the deterrence of attacks while providing continuous access. MarketNet pricing policy enhances the security of a network in several ways. First, it restrains the attacker by placing a cap on the amount a client can spend to create an attack. The extent of an attack will be limited by the attacker's available budget. Second, the access price to a resource is regulated by resource managers where, in case of a potential attack, the attacker's budget can be manipulated. Third, it controls budget expenditure of an entity by placing restrictions on how a budget is spent. A domain can direct and synchronize exposure to attacks by controlling the dissipation of currency to clients and by dynamically adjusting prices of the resources and services they offer.

MarketNet's Pricing Policy

- The total budget allocated to a particular currency domain.
- The rate at which this budget can be renewed.
- The total currency collectively allocated to all currency domains.

Unlike traditional technologies MarketNet does not rely on off-line procedures completed only hours or days after the attack. MarketNet provides resource independent-instrumentation which is continuously monitoring for attack detection by identifying anomalous spending and revenue generation behaviors. The attack detection mechanism found in MarketNet is implemented by domains through an accounting mechanism where patterns for the following are scrutinized: (a) the

spending behavior of its internal customers, (b) the request patterns of foreign domains for local currency, and (c) the revenue generation behavior of its resources. This type of data analysis provides fast information of anomalous behavior indicating a potential attack. Once an attack is detected through the correlation of gathered data the system will alert other domains and generate attack notices as well as activate protection policies.

MarketNet's protection policies, comparable to its pricing policies, impose restrictions on the power to access resources and alert its defense mechanisms. These policies are enforced by the Intrusion Response Policy Server who is responsible for activating protection policies in case of an attack. For example, the Intrusion Response Policy Server protects the system by eliminating a domain's currency. It may instantly block all further accesses by suspicious sources by configuring the bank server not to allocate any additional currency to the attackers; and by voiding currency identifiers already distributed to these sources.

Furthermore, MarketNet organizes domains in a trust management tree hierarchy to facilitate global sharing and correlation of access anomaly data, and to coordinate response to attacks. The sharing of global information is an attribute of MarketNet which does not empower the attacker to compromise and paralyze the entire network system. Through the organization of trust management in a tree hierarchy, MarketNet reports attack data to all domains on the tree-path connecting to the source. Thus, defending against multilateral attacks.

IV.3 Global Security Kernel

At the heart of MarketNet's novel technology is MarketNet Global Security Kernel, (GSK). This kernel very neatly and transparently implements and supports: a banking infrastructure , security operations, monitoring and detection components of MarketNet. The Global Security Kernel unifies and is the core of all protection logic in the transactions between clients and resources. It closely monitors the interaction between "client manager" and "resource manager" where access rights are traded in the form of currency.

Global Security Kernel Properties

The Global Security Kernel has four properties and they all contribute to the enhancement of security:

1. It allows for seamless integration with existing and future applications.
2. It allows for quantification of the power attackers can gain by attacking the infrastructure itself.
3. The GSK cannot be ambushed.
4. The accounting information maintained by the GSK uniquely associates a particular access with the corresponding entity responsible for it and cannot be tampered with.

GSK's first property provides a generic security technology for client-server models. MarketNet can be transparently applied to protect network services. The GSK's generic characteristic provides a uniform security system for heterogeneous resources, services, and clients. GSK's second property allows defenders to quantify and tune the power of an attack. For example, an attack can be controlled by depleting an attacker's budget. This can be achieved by dynamically increasing the price of a service. GSK's third property guarantees that even if part of the banking infrastructure is conquered, this will not affect other areas of the banking infrastructure. Because of the GSK's hierarchical structure the banking infrastructure can control and operate other banking facilities on the system. That is, the vulnerability of one domain does not propagate throughout the system. The Intrusion Detection and Intrusion Response Policy Server will be activated to alert other areas of the system as well as track the intruder. GSK's fourth property disempowers attackers by identifying and holding them accountable for their crimes.

The Global Security Kernel is a layer which includes modules that act as proxies on behalf of clients and resources to perform secure transactions and supports a banking infrastructure to provide all the currency related functionality, such as:

- Generation of currency.
- Exchange of currency.
- Flow of currency.
- Payment for access to resources.
- Clear and log security related transactions.

To support the above functions, the Global Security Kernel relies on the following components and procedures: currency domains, banking infrastructure, currency flow, exchange, and accounting, access management, accountability and monitoring of currency to protect resources.

(i) **Currency Domain and Banking Infrastructure**

Currency domains are the fundamental unit of protection and accountability in MarketNet. Each currency domain is represented by a bank responsible for the creation and distribution of its currency.

The banking infrastructure performs exchanges between clients and services. MarketNet's banking infrastructure is distributed and hierarchical with three different types of banks: mint banks, exchange banks, and domains banks. The hierarchical structure of the banking system supports different levels of privileges to different banks which perform different functions. For example, currency dissipation policies of subdomains are monitored and controlled by the parent banks. Banks at all levels maintain records and track all transactions.

The banking hierarchy is supported at the root level by a ***mint banks*** whose responsibility is to produce currency for all domains. Mint banks are the only banks accredited for creating currency. One of the protection policies mint banks enforce is that mint banks can generate currency for a domain only in exchange to an equal amount of currency presented by the domain bank that requests currency generation. The newly generated currency contains the name of the currency domain it is intended for, along with a unique currency identifier, and is signed by the Mint Bank.

At the next level in the hierarchy are the **exchange banks**. Exchange banks provide exchange of currencies, withdrawal and deposit functionality. Exchange banks hold the responsibility to exchange currency contingent on the dissipation policy of the domain they represent. In this step, exchange banks take the following security measure to protect against currency theft; they validate the currency received from the mint bank by signing it. While a currency is not signed by an exchange bank it is not valid. The signing of currency by the exchange bank guarantees that even if the attackers gets a hold of the mint banks' currency it will have no value. During exchanges the exchange bank will consider whether to exchange currency or not. For example, the exchange bank may have excess of one currency and decline the request.

Distribution of currency is subject to a currency distribution policy known to the resource managers. The distribution policy determines (a) whether to distribute currency to a particular client or domain, (b) the total amount and rate at which to distribute, (c) the total amount of currency distributed to all domains, and other parameters. Any transaction that takes place between banks is automatically recorded and an association is created between the two transacting entities. This association by default makes the recipient liable for the use of the currency.

Banks at the next level are the **domain banks**. Domain banks work locally within their own domain. They enforce the policies of the domains they represent and perform accounting and auditing. Domain banks also provide withdrawal and deposit functionality similar to exchange banks. Unlike exchange banks, however, domain banks are not authorized to exchange currency. Domain banks interact with exchange banks and request currency exchange on the client's behalf. For an exchange type of transaction the domain bank will approach an affiliate bank for assistance. Domain banks maintain logs that are identical with those maintained by exchange banks, omitting only the received foreign currencies log.

(ii) Currency Flow, Exchange, and Accounting

The currency flow between the client manager and the resource manager sets an irrevocable association between the client, the client's request and the bill. This association guarantees a one to one correspondence between the domain, the client, the bill, and the request. That is, a particular currency will have a unique identifier associated with a particular user in a particular domain for a particular request.

The banking infrastructure takes responsibility for the flow and usage of all currency. Bank servers monitor all transactions and provide this data to the IDM for analysis.

A typical transaction works as follows:
A client from domain X wants to access resources in domain Y. The Generic Client Manger will act on behalf of the client in acquiring the right currency to access Y's resources. The Generic Client Manger will approach X's domain bank and request currency exchange to be used in domain Y. The X bank will contact Y bank to request currency exchange. Upon receiving the request for exchange,

bank Y will check if the exchange violates any currency dissipation policies. If this is not true, it will record the transaction and return a uniquely identifiable Y currency to bank X. X bank will receive the uniquely identified currency, record the transaction, and transfer the currency to the Generic Client Manager. Having the Generic Client obtained the appropriate currency and thus conformed to the policies of the banks it forwards the currency plus the request to the resource manager to obtain a service at domain Y. In addition, the GCM embeds the signature of this request inside the payment, uniquely linking the payment and the particular request.

(iii) **Access Management**

There are two supporting components to the Global Security Kernel layer. These two components: are: the generic client manager (GCM) and the generic resource manager (GRM). The client manager and the resource manager act on behalf of the client and the server respectively. The generic client manager works as a broker between the client and the domain's bank server. The generic client manager assists clients by obtaining price information for the desired service from the service/price directory, determining the currency and amount needed to pay for the services, and contacts the client's domain bank to pay for the desired services with the appropriate currency. Having received the right currency from the local bank, the client manager forwards the client's request plus the payment to the server.

The role of the **generic resource manager** is to accept service requests and payments from the client domain. The generic resource manager entity is responsible for providing resource access to clients based on an appropriate payment. Upon receipt of a request, the resource manager authenticates the validity of received payments and deposits these payments in the target domain's bank. The resource manager also ensures that the correct payment amount has been received for the desired service, and that the received payment is in the correct denomination. Among its responsibilities, it maintains a log of accesses to the target domain and is responsible for updating price tables in the Service/Price Directory server, for use of the target domain services.

The function of these two entities, the GCM and the GRM, is to make request transactions as transparent as possible to clients and servers. Like other entities in the Global Security Kernel, the client manager and the resource manager also monitor and collect information based on the payment

of resources being accessed. This information is collected for the use of the Intrusion Detection Monitor.

The **Service and Price Directory** server's role is to maintain and update services available for a given resource domain and its corresponding prices.

(iv) **Accountability of Currency Monitoring**

The architecture of the banking infrastructure and its corresponding GCM and GRM is supported by various components that aid in: (1) assessing the performance of transactions, (2) analyzing patterns of behavior for the flow of currency, (3) detecting any potential intrusion or attacks to the system, and (4) provide economic principles to regulate demand and supply of services.

Among the components acting as security support devices to the Global Security Kernel are the Intrusion Detection Monitor and the Intrusion Response Policy Server. These two components are an integral part of the uniform and continuous security system provided by the GSK. The Intrusion Detection Monitor (IDM) is responsible for protecting its domain from attacks. The IDM provides protection to the system by collecting data through monitoring, auditing, and correlating the flow of currency through the bank's domain. Statistical algorithms are used to correlate and analyze data in search of budget spending patterns and revenue generating patterns. The analysis of currency flow in a domain can reveal anomalies and detect intrusions. An anomalous behavior, for example, could be an expenditure coming from a client who rarely uses a particular resource. An unusual allocation of budget to a client could also indicate a potential denial of service attack.

This data tracking device contains exchanges and allocation of currency by bank clients and builds a historical profile of clients. Furthermore, the banking infrastructures protects and guarantees against double spending/depositing of currency and that the currency it creates is non-forgeable.
In addition, to collecting data and scrutinizing the behavior of different clients, the Intrusion Detection Monitor can also identify and isolate an attacker. The Intrusion Detection Monitor will use the currency identifiers to determine the source domain and owners of the currency used in the attack. Based on the information collected by the IDM, it will determine if the IRPS should be alerted.

The **Intrusion Response Policy Server** (IRPS) is responsible for activating intrusion protection policies in a system. In the event the Intrusion Response Policy Server is alerted by the Intrusion Detection Monitor, it will take action according to the severity of the attack. The IRPS can activate various protection policies. These policies will control the access by distrustful domains. For example, one of the policies implemented by the IRPS is to configure the bank server so that the intruders accounts are blocked immediately. Another security measure the IRPS can activate is the revocation of currency identifiers already distributed to the suspicious entities.

The Intrusion Detection Monitor provides MarketNet with defenses that is resource and service independent. Unlike traditional protection mechanisms which use binary access control, and are unable to distinguish between and internal or external users MarketNet's monitoring system provides a uniform and continuous access control. Resources are instrumented to use currency for access control and monitoring. The currency flows provide a uniform means to express and monitor access behavior.

V. APPLICATION OF MARKETNET TO SNMP

In this section we will first discuss how MarketNet can protect SNMP from attacks. Then, we will examine a protection strategy for SNMP's vulnerable port number. SNMP is a likely candidate to be enhanced with MarketNet's security system because it represents the typical client-server models found in network services, and because SNMP is a network service with critical security risks, i.e., SNMP is used for setting network configuration variables. A successful attack on these variables can compromise a whole network. SNMPv3's security technology exhibits the traditional vulnerabilities found in applications which use binary access, ad-hoc programming, and customized software. The security technology implemented in SNMPv3 is a subset of the technology offered by MarketNet. Unlike SNMPv3, which is customized to protect SNMP engines only, MarketNet provides global protection against threats by monitoring access to its resources through the use of currency and paid fare.

SNMP's framework is composed of a manager (the client), an agent (the server), plus a transport (TCP/IP, UDP) mechanism where communication takes place between client and server, see figure 3, below.

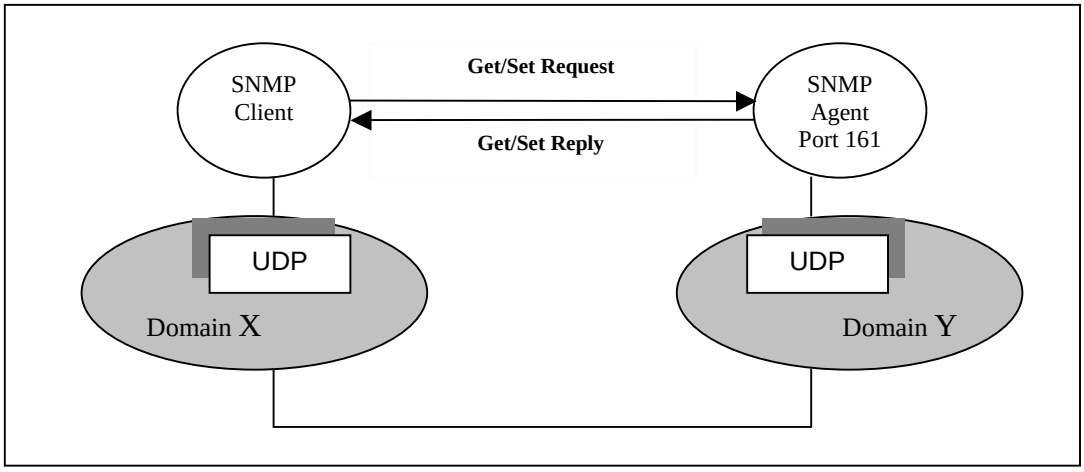


Figure 3: SNMP Architecture

A major advantage in using MarketNet to create a secure SNMP environment is that it can be easily implemented. MarketNet provides transparent protection without any need for modification. No changes are needed to either the SNMP client or SNMP server.

In fig. 4 below, depicts MarketNet as applied to SNMP. The Global Security Kernel, is inside the dotted line box and oversees the activities of its domain and protects the services it provides. The Global Security Kernel generates and controls the distribution of currency and enforces the policies of its usage. This currency system is designed with the intention to protect the domain against assaults. The currency generated by the banking infrastructure contains a unique identifier that allows the system to track the identity of an attacker. The currency's unique identifier also provides authentication to protect against frauds, such as, forgery of currency and double spending or deposits. Because the banking infrastructure is a desirable target for assault, it transacts through secure protocols and these transactions are closely monitored by the mint bank. MarketNet transacts its currency through secure protocols using cryptographic functions to provide privacy and authenticity.

Implementation of MarketNet on SNMP's architecture is accomplished by connecting SNMPs client server to the generic client manager server and the agent to the generic resource manager. SNMP like MarketNet is a client-server application. When applying MarketNet technology to protect a network system such as SNMP, the following two components, the Generic Client Manager and the Generic Resource Manager, act as proxies between the SNMP client and SNMP server, respectively (see fig. 4).

MarketNet's GCM and GRM servers act as validation check points in order to provide access control and access control management to SNMP services. The GCM and GRM are responsible for interpreting the SNMPs PDUs and the original PDUs are not altered. SNMP PDUs travel from client to server in a transparent manner to the user.

The GCM receives requests and scrutinizes their validity. It is responsible for accepting or rejecting a message request. In the case of insufficient payment, requests will be discarded. Once the generic client manager does its record keeping and security check list, it approaches the global security kernel.

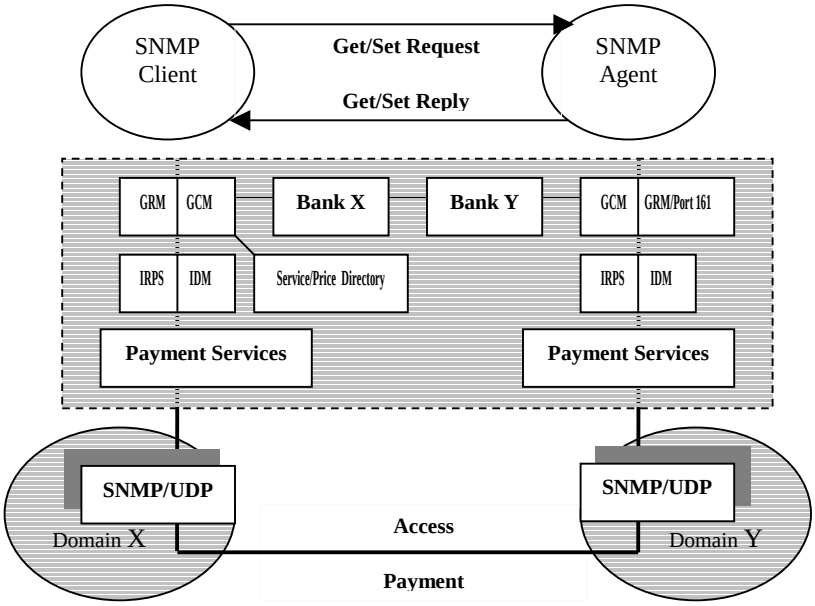


Figure 4 : SNMPvM Architecture

A client's request for service is processed as follows.

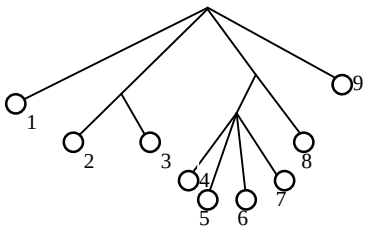
1. Access to acquire SNMP service is initiated by the SNMP client making a request.
2. This request is received and processed by the Generic Client Manager. The Generic Client Manager performs its routine of checking price and availability with the Service/Price Directory server. Based on the information obtained from the Service/Price Directory, the GCM will contact the client's bank server to obtain the currency to pay for the service.
3. The local bank proceeds to obtain the funds for the transaction. The local bank will be responsible for obtaining the appropriate amount in the currency accepted by the resource domain. The local bank will also determine whether currency exchanges are necessary. If the client's request can be satisfied, that is budget and services are permitted, then GCM constructs a payment voucher PDU.
4. This PDU is forwarded to the Generic Resource Manager with security measures. The payment voucher PDU contains the actual payment, the signature of the original SNMP request, the identifier for the manager process, plus a time stamp. These values, the process

identifier, the signature of the original request plus the time stamp are used to determine the validity of the transaction.

5. The GCM will send a second message to the Generic Resource Manager containing the original SNMP request plus the signature of the SNMP request.
6. The Generic Resource Manager receives both these packets and reconciles them. It performs authentication by computing the signature of the SNMP payment voucher with the signature attached to the original SNMP request.
7. If the match is successful the GRM proceeds to verify prices for the request being processed. If the price agrees with the received payment the GRM dispatches the service request to the SNMP server to provide the paid and authenticated for services. It also credits the server's bank account accordingly.
8. The GRM gets a response from the SNMP server and sends a reply to the GCM.

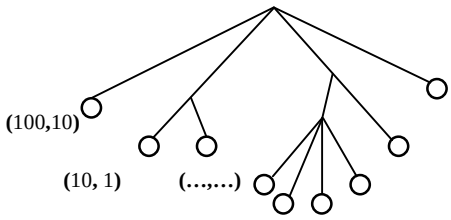
In addition to the above implementation of MarketNet to SNMP, the resource manager which protects SNMP was enhanced with a pricing module. The purpose of this module is to maintain a pricing database to support pricing and protection policies for SNMP. This pricing database is based on the concept of an SNMP MIB, but unlike the original SNMP MIB which stores values of variables, the pricing MIB stores prices to access the variables. That is, a database where prices can be dynamically maintained and edited. The pricing MIB is maintained by the resource manager. Following is an illustration of an SNMPvM pricing MIB and how it works.

Original MIB



Figures 5a: SNMP MIB

Pricing MIB



Figures 5b: SNMPvM Pricing MIB

The pricing MIB works as a mirror image of the original MIB. The original MIB contains the value for network configuration variables and the pricing MIB contains the cost to access these variables. The pricing MIB has a one-to-one correspondence with the values stored in the original MIB. On the pricing MIB, the left side of the parenthesis denotes the price for SET (modify a variable) and the right side GET (read a variable). For example, the value for using SET on variable O_1 (see fig. 5a) has the price of 100 units. This is the price a client would pay to SET O_1 variable. The idea of the pricing MIB is to be able to support pricing policies and protection policies. Prices in the pricing MIB can be dynamically changed by the resource owner. For example, if a client's activities are considered suspicious these prices can be increased in order to accelerate the client's budget expenditure. Conversely, in order to promote a service the resource manager can decrease a price.

While MarketNet can protect SNMP from different types of attacks, SNMP's well known port 161 is a vulnerable spot. An attacker can access port 161 directly and bypass MarketNet's protection. Ports or portals used in computers, give an easy entry point of access to intruders into a network. Servers with a listening and waiting port to communicate are the perfect place to mount an attack. An ill-intentioned client-computer can grab this port, which is open and listening for connections, and create a potential point of attack. In this project we protect port 161 by using the following port security strategy.

The strategy implemented in this project to deter intruders from attacking port 161 follows:

1. conceal SNMP's true port number.
2. transfer the real SNMP to an undisclosed machine.

SNMP's true port number was concealed by simulating a connection to SNMP with a preset port number. The preset port number acts as a disguise leading not to the services of the true SNMP but to an authenticator/server that is responsible for verifying the legitimacy of the issuer of the request. The authenticator/server works as a check point and filtering device. Through a minor modification to the SNMP code itself, requests not coming directly from the generic resource manager are filtered out. The transferring of the real SNMP to an strategic machine only accepts requests coming from the generic resource manager.

VI. SECURITY COMPARISONS BETWEEN SNMPv3 AND SNMPvM

As we have seen in section III, SNMPv3 was created to provide security features to support SNMPv1 and SNMPv2. However, when comparing the security technology implemented in SNMPv3 to the security technology implemented in MarketNet, it is evident that SNMPv3 does not meet the standards needed to protect itself in a network system. The main reason for this deficiency is that SNMPv3 relies on traditional security technologies, as discussed in Section IV, where the lack of a uniform security mechanism and the lack of continuously monitoring access control to resources contribute to a vulnerable network system.

The first advantage offered by MarketNet in contrast to SNMPv3 is its ease of use. MarketNet provides enhanced security over SNMPv3 and can be used to protect SNMPv1 and SNMPv2 from attacks. While MarketNet can be readily applied to SNMPv1 and SNMPv2 to provide security protection, SNMPv3 requires changes to either the protocol and to the MIB structure. MarketNet affords a transparent protection without the need for modification. The security operations carried out by the generic client manager and the generic resource manager, on behalf of their clients and resources, are accomplished seamlessly and without requiring changes to either the protocol or Management Information Base (MIB) structures. Furthermore, MarketNet neither modifies the SNMP server nor the SNMP client. MarketNet's ease of use and transparency in itself provides extra security since it does not expose the system to additional vulnerabilities created by any new components. Unlike MarketNet, SNMPv3's security mechanism is not generic and is packaged as a specific part of the message and can only be used on SNMP engines.

In addition to providing transparency and ease of use, MarketNet provides all the security features of SNMPv3 and more. The following security enhancements are key elements granted in MarketNet which limit and control the power of attackers. MarketNet empowers the defense of network systems by controlling the exposure to attacks, by holding clients accountable for attacks, and by its ability to detect any suspicious elements. None of the following security strategies are implemented in SNMPv3:

1. Accountability for use of resources.
2. Detection of attack source.

3. Quantification of exposure to attacks.

Holding the users responsible for their actions is an important security measure in preventing attacks. Accountability is a deterrent for attackers as the goal of an attacker is to be able to exploit a system without being detected or identified. SNMPv3 does not implement a security technology which holds its clients accountable for the use. SNMPv3 cannot monitor its client's use of resources nor can it monitor the amount by which this resources is used. Without accountability an attacker can repeatedly strike at SNMP since there are no consequences for the crimes perpetrated. Conversely, MarketNet can determine specifically how clients are using resources. MarketNet has the technology to detect suspicious activities and can immediately identify it and isolate it. Every transaction exchanged between the client and a service is irrevocably associated with the issuer of the transaction, the currency used for the transaction, and the request for that particular transaction. MarketNet's accountability system provides security by holding attackers responsible for their actions.

Intrusion detection is a second important security feature lacked by SNMPv3. SNMPv3 is not capable to detect dangerous intrusion attempts. Once an intruder successfully gains access to the system, the intruder is free to attack and the system is rendered helpless. There is no defense mechanism to stop an attack or control the damage caused by one. SNMPv3 makes itself vulnerable by assuming that the network, where its security policy is implemented is not malicious. SNMPv3 does not have a mechanism to check or verify anything about the source where the request originates. SNMPv3 does not maintain a history on its clients and thus cannot evaluate the type of clients its servicing. This is an ideal scenario for attackers who wish to exploit trust among hosts. SNMPv3 makes the assumption that the "bad guys are outside." In contrast, MarketNet's logic is that the "bad guys" could very well be inside. MarketNet is implemented to work with distrustful entities and its defense mechanism provides the ability to monitor the kind of activities a client is engaged. Each client that MarketNet services is continuously monitored. Every request and transaction made by clients is recorded and evaluated by the Global Security Kernel. SNMPv3 does not implement this type of technology.

A third and important security consideration completely absent in SNMPv3 is the system's ability to control the exposure to attacks. As said previously, once the attacker is granted access the system is

rendered helpless. Being able to quantify exposure to attacks proves to be a highly valuable security feature because it allows the defender to minimize its losses. MarketNet can effectively control an attack. It can apply its pricing policies or its intrusion detection policy. Using pricing policies, MarketNet can exacerbate the expenditure of an attacker by increasing the price of the resources and depleting the attacker's budget. For example, a Denial of Service attack is where legitimate access is prevented by bombarding a resource with requests. MarketNet's "quantification of exposure to attacks" security feature provides readily protection against this attack. The resource manager will control both budget dissipation and prices, thus limiting the budget available to the attacker. Unlike SNMPv3, it has no protection against Denial of Service.

An example which illustrates SNMP's security deficiencies is the stealing of passwords. This one example alone demonstrates how accountability for use of resources, intrusion detection, and quantification of exposure to attacks render SNMP exposed. A stolen password is all an attacker needs to conquer SNMP and paralyze its activities. First, the attacker is not exposed, nor will he/she be liable for the crimes committed since its identity will be unknown. Accountability for use of resources is not implemented in SNMPv3. Second, since the intruder has stolen the identity of a legal user, the unsuspecting system cannot infer that it is under attack. At this point, SNMP solely relies on the trustworthiness of the client. And third, SNMP can neither diminish the intensity of the attack nor stop it. The detection and assessment of an attack will occur after the facts. These major vulnerabilities found in SNMPv3 are corrected in SNMPvM, where access is acquired through currency. The currency and prices available to a client are controlled by the banking infrastructure and the resource mangers who continuously evaluate the amount of access rights a client should have to access resources. An anomalous behavior on the client's behalf will be detected and questioned.

In addition to the securities described above, MarketNet offers comparable security in the areas of:

- modification threat.
- masquerade threat.
- eavesdropping threat/stream modification threat.

Like SNMPv3, MarketNet provides protection against modification threat. As explained in Section V, first, MarketNet uses the payment voucher to embed the signature of the request, then it sends a

second message with the signature of the SNMP request. The two signatures are reconciled by the Generic Resource Manager and authentication is achieved with a successful match. Masquerade threat is handled by MarketNet through non-forgeable payment which uniquely identifies the origin of the currency. Similar to SNMPv3, MarketNet handles disclosure threat and message stream modification by using cryptographic functions.

VII. CONCLUSION

Although a high level of network system security was achieved on SNMPv3, its design is limited. SNMP's hard-wired security system can solely be used on SNMP, plus modifications are needed when used to protect versions 1 and 2 of SNMP. Furthermore, SNMP remains unprotected in the areas of accountability, ability, and attack quantification. Thus, leaving potential points of attacks open to ill-intentioned clients. MarketNet, on the other hand, is a generic security system that can be easily applied to any client-server application such as SNMP. MarketNet is a dynamic and versatile security system. The damage attackers can create on a system protected by MarketNet is limited since the domains dynamically defend themselves at the signs of abnormality. If a potential attack is detected MarketNet will use its security capabilities to corner the attacker: it will block resources, limit the spending power of suspicious clients, and control the pricing of resources.

Foonotes:

---

1. A. Dailianas, Y. Yemini, D. Florissi, and H. Huang, “MarketNet: Market-based Protection of Network Systems and Services-an Application to SNMP Protection,” Computer Science Department, Columbia University, New York, New York.
2. Dailianas, A. “MarketNet: Use of Currency for Access Control in Large-scale Information Systems,” Ph.D. Thesis, Department of Computer Science, Columbia University, 2001.
3. Dailianas, A. “Use of Currency for Access Control in Large-scale Information Systems,” Ph.D. Thesis Proposal, Department of Computer Science, Columbia University, 1998.
4. W. Stallings, SNMP, SNMPV2, SNMPV3, and RMON1 and 2. Reading, MA. Addison-Wesley, 1999.