MarketNet: Market-based Protection of Network Systems and Services – an Application to SNMP Protection

A. Dailianas, Y. Yemini, D. Florissi, and H. Huang

Department of Computer Science
Columbia University,
450 Computer Science Bldg.
New York, NY 10027
Email: {apostolos, yemini, df, hhuang}@cs.columbia.edu

Abstract - **Current networks allocate attack and protection powers asymmetrically. Attackers can exercise virtually unlimited power to attempt to compromise systems and evade detection and accountability, while the owners of these systems are mostly limited to developing passive protections and keeping up with new attack techniques. This paper describes novel protection technologies, developed by the MarketNet project at Columbia University, that shift power from attackers to defenders, giving the defenders control over the exposure to attacks and over detectability and accountability of attackers. MarketNet uses market-based techniques to regulate access to resources. Access to a resource must be paid-for with currency issued by its domain. Domains can control the power of attackers by limiting the budgets allocated to them, and control the exposure of resources by setting their prices, effectively providing a quantifiable access control mechanism. Domains can monitor currency flows and use uniform resource-independent statistical algorithms to correlate and detect access anomalies indicating potential attacks. Currency is marked with unique identifiers that permit domains to establish verifiable accountability in accessing their resources. Domains control and fine tune their exposure to attacks; adjust this exposure in response to emerging risks; detect intrusion attacks through automated, uniform statistical analysis of currency flows; and establish coordinated response to attacks. MarketNet mechanisms unify and kernelize global information systems protection by containing all protection logic in a small core of software components.**

The paper presents the architecture and operation of MarketNet along with the design and implementation of main architectural components. The paper illustrates the application of MarketNet to the protection of the Simple Network Management Protocol (SNMP) and compares it with the security features offered by SNMPv3.

I. INTRODUCTION

Protecting large-scale information systems and guaranteeing availability remains an elusive challenge of ever-growing importance and complexity. Exposure to insecurities and the opportunities available to attackers is increasing with the growth in the range of resources, and with the scale, complexity, and operations management practices of different domains. Traditional protection technologies have focused on shielding target resources; they involve ad-hoc instrumentation to monitor resource access and manual correlation of these access logs to detect intrusion. Protection software monitors and filters untrusted accesses to resources and manages trust relationships among users and resources. Attackers can pursue virtually unlimited attempts to compromise a system and are typically left unaccountable for abuses and crimes that they commit.

Rapid changes in technologies increase the vulnerability to attackers. Current protection technologies are specialized to each component. A minor insecurity in a new component can propagate substantial exposure to other components. Thus, insecurities can be formed non-monotonically; i.e., a system that is secure can be rendered insecure by the addition of a single component. The combinatorial explosion of interactions between new components and existing ones creates ample possibilities for insecurities. In the absence of a unifying security architecture it is nearly impossible for component vendors or domain administrations to accomplish coordinated protection.

Domain administrations are thus increasingly exposed to security risks and are unable to control, or even assess this exposure. They usually resort to expert manual labor to monitor and correlate access anomalies and detect an attack, typically through off-line procedures completed only hours or days after the attack. Even when an attack is detected, identifying the responsible source requires complex ad-hoc collaborations of experts (and in many cases is unfeasible). The potential for attacks and the complexity of protection increases with each change in resources or configuration.

MarketNet enables domains to quantify and dynamically tune their exposure to attacks. Resources and clients are organized in currency domains. Resources are instrumented to use currency for access control and monitoring. Clients wishing to access a target resource must pay resource managers the access price with the currency of the target domain.

This research is sponsored in part by the USAF, Air Force Materiel Command, under contract F30602-97-1-0252, "MarketNet: A Survivable, Market-based Architecture for Large-scale Information Systems", and in part by the New York Science and Technology Foundation, subcontract with Polytechnic University. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of the Defense Advanced Research Projects Agency (DARPA), the Air Force, or the U.S. Government.

Currency dissipation is entirely under the control of domains. Possession of currency of the target domain authorizes a client to access resources in the target domain. The amount of currency of the target domain available to a client – or an attacker – limits its ability to gain access to resources and to cause damage. Domains control and tune exposure to attacks by controlling the dissipation of currency to clients and by dynamically adjusting prices of the resources and services they offer.

MarketNet provides continuous – as opposed to binary (yes or no) – access control. The price of a resource, controlled by the resource manager, along with the budget available to clients, defines a dynamic and flexible access control discipline to resources. The access control can take into account variations in the supply and demand, security considerations of the resource managers, or the relative importance of resources to different users.

MarketNet enables rapid identification and isolation of attack sources. Currency carries unique non-forgeable identifiers used to establish undeniable accountability for access to resources. When an attack is detected, currency identifiers reveal the source of the attack. The target domain can prevent further abuses by eliminating new budgets or voiding currencies, effectively isolating the attack source.

MarketNet provides resource-independent instrumentation to detect attacks and enables correlation for attack detection and coordination of response to attacks. A domain maintains accounting information of (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 accounting provides uniform, resource-independent instrumentation of access and forms the basis for attack detection by identifying anomalous spending and revenue generation behaviors. 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. Traditional statistical analysis algorithms are used to correlate the data and detect access anomalies indicating attacks. The target domain reports attack data to all domains on the tree-path connecting to the source. Each domain correlates the data of its subordinates to analyze global attack patterns and reports the results to all of its subordinates. Subordinates use this global correlation to enhance local detection and response facilitating detection of multilateral attacks and coordinating responses.

MarketNet mechanisms are scalable and enable protection among mutually distrustful domains organized in a large scale federated system. Furthermore, the protection mechanisms are entirely independent of the underlying resources, thus can be be retrofitted into an existing system with minor adaptation of its components.

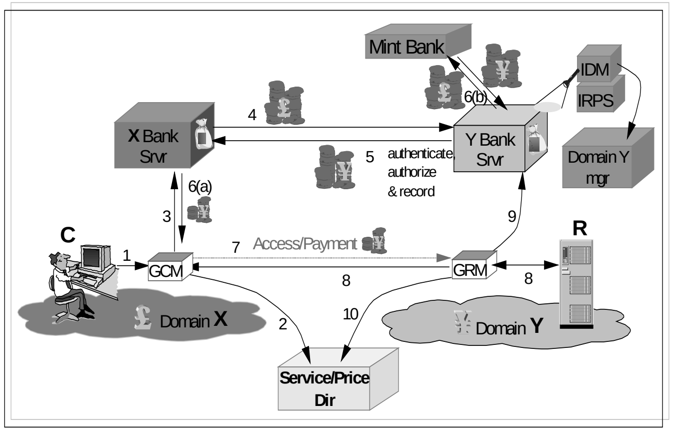

Fig. 1: Operations in typical access transaction

The remainder of the paper is structured as follows. Section II presents the architecture and operations of MarketNet, and demonstrates how MarketNet can guard against several types of attacks. Section III shows how MarketNet can protect Internet applications and, in particular, SNMP. Section IV discusses related work.

II. MARKETNET OPERATION AND ARCHITECTURE

A. *Operations Model*

MarketNet builds around the MarketNet Secure Kernel (MSK), a core of protection agents, mechanisms, and protocols. The typical access transaction in MSK is depicted in Fig. 1. Client application *C* in source domain *X* wishes to access resource *R* in target domain *Y*.

The steps in the transaction are:

Step 1. The client *C* invokes the local Generic Client Manager *(GCM)* to mediate the access.

Step 2. The *GCM* accesses the Service/Price Directory server to acquire the price of *R*.

Step 3. The *GCM* needs to acquire enough amount of currency of the target domain *Y* to pay for *R*; it approaches the bank server of its domain *X* to obtain it.

Step 4. The *X* bank server contacts the *Y* bank server and requests currency exchange. The *X* bank server may request more currency than needed by *C* and cache it for later requests.

Step 5. The *Y* bank server validates the received *X* currency. It checks if the exchange violates its currency dissipation policy. If not, it records the transaction and passes a uniquely identifiable *Y* currency to the *X* bank server.

Step 6(a). The *X* bank server passes the *Y* currency requested by *C* to its *GCM* and records the transaction.

Step 6(b). The *Y* bank server exchanges the *X* currency with its parent mint bank to acquire more *Y* currency for the *X* currency it deposits.

Step 7. The *GCM* of *C* uses this currency to pay the Generic Resource Manager (*GRM*) of *R* for accessing *R*. The GCM embeds the signature of this request inside the payment, uniquely linking the payment and the particular request.

Step 8. The *GRM* of *R* monitors, logs accesses to, collects, and validates payments to *R*.

Step 9. The *GRM* of *R* deposits revenues in its account with the *Y* bank server.

Step 10. The *GRM* of *R* updates the price for *R* to reflect its current availability.

The Intrusion Detection Monitor (*IDM*) examines and audits the flow of currency through the bank server. It uses generic statistical algorithms to detect and analyze access anomalies. For example, an unusual allocation of budget to client *C* to access the resource *R* may indicate a potential denial of service attack. Similarly, anomalous expenditures on a rarely used Application Program Interface (API) may indicate an attack pattern. In addition, the *IDM* can quantify the severity of potential attacks in terms of the resource price, its capacity to generate revenues, and the excess budget available to the suspicious attacker. The *IDM* also analyzes the audit data to identify the sources of the attacks. It uses the currency identifiers to determine the source domain and owners of the currency used in the attack. It generates attack notifications to the *MSK* Domain Manager as well as the Intrusion Response Policy Server (*IRPS*).

The IRPS may activate various protection policies. In particular, it may instantly block all further accesses by suspicious sources by configuring the bank server not to allocate any additional currency to them; and by voiding currency identifiers already distributed to these sources. Other intrusion responses may include detailed analysis of bank server audit data logs, reports to the banking system of the attack data, and local corrective actions. The banking system correlates the attack data to identify and shut domains responsible for global coordinated attacks.

B. *MarketNet Security Kernel (MSK): Architecture and Implementation*

The overall architecture of MSK is depicted in Fig. 2. This section explains the function and key implementation characteristics of each component.

The *bank server* maintains accounts and scrutinizes the currency flow of a given domain. Access rights are traded in the form of currency that gives the holder the right to access resources in the respective domain. Each domain has its own currency and currency dissipation policy controlled and enforced by their bank server. The banking infrastructure is therefore certainly very attractive to attackers and their design has to be resilient to such vulnerability. Bank servers maintain accounts for all GRM and GCM and support secure debit/credit transactions with respective accounts. They use secure protocols to exchange currencies with other bank servers. Each bank has strict control over the amounts of currency it exchanges. Therefore, even if an attacker somehow manages to acquire infinite amounts of foreign currency, s/he will not be able to exchange it for the desired target domain currency. Bank servers collect audit trails of all transactions (including exchanges and allocation of currency to customers) to be used by the IDM for monitoring, identification, and isolation of source of access. For scalability, protection, and administrative reasons, bank servers are organized hierarchically with the *mint bank* at the head of the banking hierarchy.

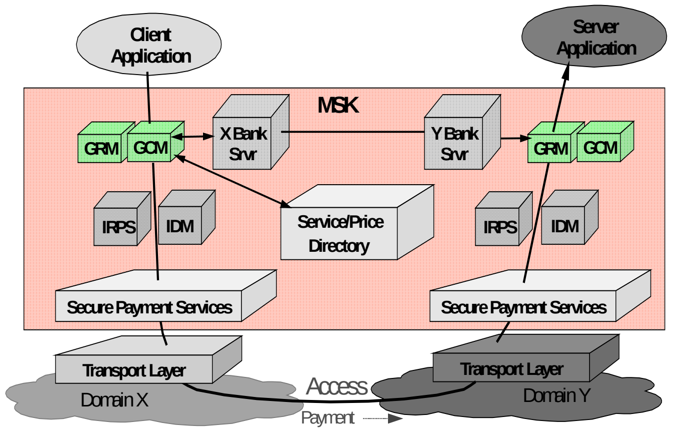

Fig. 2: Architecture of MSK

The MSK implementation of the banking infrastructure has to be efficient and secure, while accounting for the flow of the access rights. The current prototype implementation of the banking infrastructure focuses on creating currency with unique identifiers, detecting double spending/deposit, guaranteeing that currency cannot be forged, exchanging currencies securely, and imposing limits on the total wealth an intruder can accumulate even if successful in taking over a bank. In order to limit time-consuming currency exchanges, banks cache foreign currencies in their local reserves in anticipation of future demand by local GCM. The detailed description of the implementation details used by MSK is beyond the scope of this paper, but can be found in [1].

The *GRM* provides clients with access to a resource given enough payment. Its role is to make the MSK as transparent as possible to clients and servers. It monitors and accounts for usage of resources by GCMs, extracts the respective payment; supports a secure GCM-GRM payment protocol, contacts the local bank to deposit revenues generated by the resource being managed, and finally (in cooperation with the local MSK domain manager and the IRPS) updates resource prices to reflect the current operating conditions of the resource (such as load) and security considerations.

The *GCM* manages client accesses to resources and performs payments. In analogy to the operation of the GRM for the resources, the GCM abstracts the necessary payment and budget accounting related functionality to make the MSK as transparent as possible to client applications. The GCM supports acquisition of price information and respective resource selection subject to the client application quality of service constraints and budget allocation and handles acquisition of currency and disbursement of payments to GRM through a secure GCM-GRM payment protocol.

The *service/price directory server* maintains service/price information for domain resources and implements query protocols for MSK services such as the GCM, the GRM, and the IDM. In particular, the directory supports a secure price advertising and updating protocol for the GRM.

The *secure payment services* layer enables a variety of payment options. The GCM selects the one that best suits the needs of applications. The current prototype implementation has three alternative options. The first sends the payment to the server on a separate packet that precedes the actual request. The signature of the request is embedded in the payment packet to avoid any malicious association of the payment with a different request. The second option attaches (transparently to the client) payment in a special header in the packets sent to the server. The third option is similar to the second, but adapted for real-time applications such as video on demand. The client-side periodically sends very small amounts of currency to the server paying for the continuous service the client receives, adding minimal overheads to the overall transaction. The secure payment services layer can be extended to include other options that take advantage of underlying technologies such as Ipsec [2].

The *IDM* monitors, audits, and correlates currency flows to detect attacks. It supports uniform statistical algorithms to correlate and analyze audits and detect anomalies. The IDM monitors anomalies both in the budget spending patterns of clients and in the revenue generated by resources. It further correlates the revenues generated by different resources and the budget spent by different sources to detect attacks that target a collection of resources or ones that are initiated by a coalition of seemingly unrelated attackers. The IDM identifies and isolates the source or coalition of sources responsible for an attack. Finally, the IDM computes the severity level of the potential attack through a cost model that reflects the importance of attacks using metrics such as the loss of revenue as a result of an attack. Upon detection of an ongoing potential attack, it notifies the Domain Manager, the IRPS, and other IDMs.

The *IRPS* executes policies to control access by suspicious domains. It monitors attack notifications and classifies them according to their severity and the domain security policies. It automatically activates response policies to various attack classes, including notification of the attacking domain authorities. The IRPS supports control of budgeting that is reflected in the currency distribution policy of the domain. It further suggests price adjustments that are intended to drastically reduce the power of attackers and to reveal attacks that are hidden among legitimate accesses.

III. PROTECTING INTERNET SERVICES WITH MARKETNET

MarketNet is a generic solution for protection of resources and services, such as processor cycles, memory, disk space, operating system services, web services, network services, etc. Prototype implementations of all the components outlined in Section II have been developed. MarketNet has already been experimentally applied to a variety of Internet services with minimal, if any, changes to the services. The focus of this section is to give examples of applying MarketNet to protect network services. Application of MarketNet in other domains is the subject of future publications.

The following example shows the MarketNet-instrumented version of the Simple Network Management Protocol (SNMP). SNMP has been chosen to exemplify two aspects of MarketNet. First, MarketNet can transparently protect existing services without modifying them. SNMP is a typical client-server model representative of many existing Internet services and, thus, the methodology and conclusions presented in this section are valid for all of such services. Second, the current development of a customized security version of SNMP [3] gives us an opportunity to compare the security offered by MarketNet – a generic solution applicable to all Internet services – with the security offered by the SNMPv3 [3, 4] solution customized for protecting SNMP.

Work is underway that will investigate the application of MarketNet to protect other network service that can better expose the powerful MarketNet concepts (such as Web servers).

A. *Protecting SNMP With MarketNet*

SNMP is a protocol to manipulate values of objects of devices in a network, which may include contents of hardware registers, software variables, etc. SNMP queries typically include two operations: GET and SET. GET requests the value of a particular object and SET assigns a value to an object. Early versions of SNMP (SNMPv1) could be used to attack resources and services by manipulating configuration data in devices because they did not incorporate protection mechanisms. This presented a significant security risk.

The SNMPv2 Working Group has unsuccessfully attempted to introduce a standards-based security framework to SNMP. Two versions of SNMP v2 – SNMPv2u and SNMPv2* [5, 6] – incorporated security, but it was too complex, incompatible with SNMPv1, and lacked the endorsement of the Internet Engineering Task Force (IETF). The SNMPv3 Working Group [4] was chartered to produce a security framework based on the convergence of the main concepts of SNMPv2u and SNMPv2*.

Despite great care in designing these variants to ease migration and compatibility, they have not been successful yet in capturing market share. MarketNet may be used to secure SNMPv1 or SNMPv2 without requiring changes to either the protocol or Management Information Base (MIB) [7] structures. It is important to notice that MarketNet protects SNMP without any changes in the SNMP server or client applications. The GCM and GRM may be viewed as proxies between the SNMP client and server responsible for translating plain SNMP PDUs into MarketNet aware PDUs. The original SNMP operation, PDUs, MIBs, clients, and servers are not changed.

Fig. 3 depicts the standard structure of the SNMP service and SNMP protocol, which uses UDP transport. SNMP allows read (get, getnext) and write (set) operations by managers (clients) from/to network configuration objects stored in devices or files. The SNMP agent (server) provides the instrumentation for access to these configuration objects. The SNMP daemon waits on the well-known socket port 161 for service requests from SNMP client applications (get, set, getnext, etc.). Upon arrival of a request, the daemon services the query and returns the result or error to the client. Notice that the figure implies that the SNMP daemon may be running on one or more designated hosts of Domain Y, which can provide manipulation of configuration objects pertaining to multiple devices and resources controlled by each host. In versions 1 and 2 of SNMP it was relatively easy for an attacker to fake the source of SNMP requests and perform illegal configuration changes or acquire some private configuration information. The latest version of SNMP – SNMPv3 [3] – is designed to overcome the security shortcomings of earlier versions.

Fig. 4 shows the structure of secure SNMP via MarketNet (SNMPvM). The MarketNet proxies GCM and GRM protect the interactions between the SNMP servers (agents) and the clients (managers). The SNMP client sends its request through the GCM, which operates as described in TABLE I.

The GRM proxy at the SNMP server listens to port 161 and receives all requests for the SNMP server. Upon receiving a message from the GCM it operates as described in TABLE II.

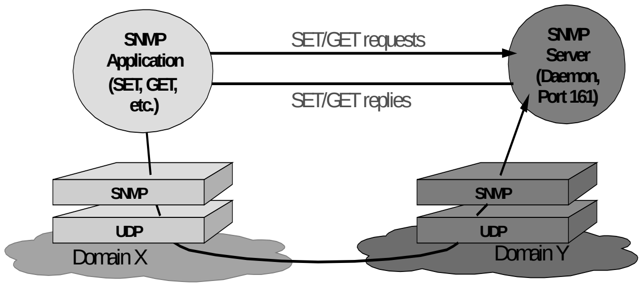
SNMP Application (SET, GET, etc.)
SET/GET requests
SET/GET replies
SNMP Server (Daemon, Port 161)
SNMP
UDP
Domain X
SNMP
UDP
Domain Y

Fig. 3: SNMP Service Structure

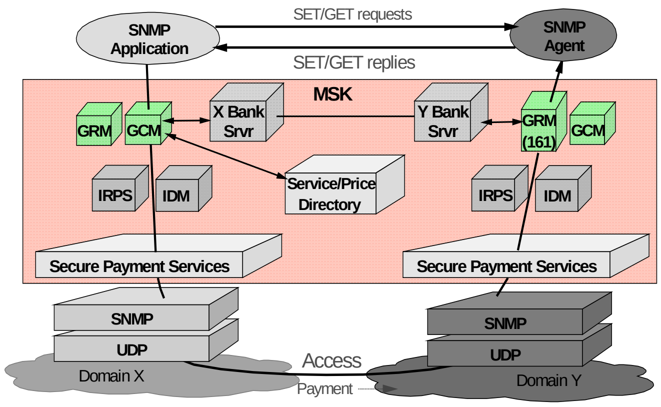

Fig. 4: Structure of SNMPvM

TABLE I
The MarketNet Client (GCM) Protocol

1.	Obtain the price and currency for serving the request by querying pricing information at the server.
2.	Obtain the appropriate budget through the local bank server or use cached budget. (To complete this operation, the local bank server may have to contact the SNMP client bank server to perform currency exchange).
3.	Construct the payment PDU and send it to the GRM at the server. The payment PDU consists of the currency, the signature of the original SNMP request, the manager process identifier, and the current timestamp. The payment PDU uniquely links the payment, the original request, and the specific manager process.
4.	Send the original SNMP request to the GRM at the server.

TABLE II
The MarketNet Server (GRM) Protocol

1.	Extract the payment and signature from the payment PDU. Compute the signature of the SNMP request and match it with the respective payment signature. If requested, check that the request has not been excessively delayed.
2.	Compute the price of each component of the request using a local price MIB.
3.	Pass to the SNMP server those parts of the SNMP request that can be serviced with the payment extracted in step 1.
4.	Obtain response from the server. Package the response and send to GCM.

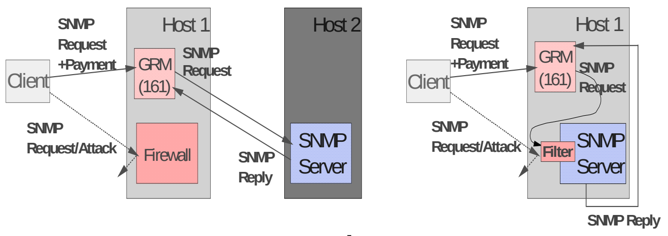

Fig. 5: Protecting the SNMP Server

The scheme has the vulnerability that the attacker may discover the port in which the actual SNMP daemon is running and bypass MarketNet security by directly accessing that port. Fig. 5 depicts the interactions discussed and presents a couple of alternative ways to deal with attacks on the real SNMP server.

On the left side of Fig. 5 the SNMP server is transferred to another machine. A firewall filters any request that does not come from the GRM. Notice that there is no need for complete firewall functionality. Minimal filtering capabilities are enough to provide the desired protection. On the right side of Fig. 5 the filtering of requests is performed either through support from the operating system (e.g., in Linux the kernel can be configured to apply specific filters to the incoming traffic) or through a minimal modification on the SNMP server code itself to reject any request not coming from the GRM. In both alternatives, the server sends its replies directly to the original client. Attackers that try to send requests directly to the SNMP server are filtered by the site firewall. Attackers that send requests with inappropriate payment are discarded by the GRM.

B. *Comparison of security features of SNMPv3 and SNMPvM*

TABLE III gives a brief summary of the security features found in SNMPv3 and SNMPvM. The kinds of protection presented in TABLE III are taken directly from the SNMPv3 Internet draft [3].

In addition to the security features presented in TABLE III SNMPvM offers all the security features provided through the instrumentation with MarketNet, such as the ability to trace back requests, to provide accountability in the manipulation of SNMP related information, to isolate the sources of attacks, the possibility to price each SNMP object and access differently, etc. Furthermore, prices and the budgets available to clients can be used to prioritize access during periods when security considerations or congestion dictate limiting access to only a subset of clients.

TABLE III
Security in SNMPv3 and SNMPvM

SNMPv3	SNMPvM
Immunity against the *modification threat*, i.e., the danger that some unauthorized entity may alter in-transit SNMP messages generated on behalf of an authorized principal in such a way as to effect unauthorized management operations, including falsifying the value of an object.	*Equivalent protection*. The payment for the execution of a specific operation includes a signature of the request as part of the payment. This signature is compared against the signature of the received request preventing unauthorized alterations of messages in-transit.
Immunity against the *masquerade threat*, i.e., the danger that management operations not authorized for some user may be attempted by assuming the identity of another user that has the appropriate authorizations.	*Completely avoided*. The request is accompanied by the appropriate non-forgeable payment that uniquely identifies the requester.
Immunity (when necessary) against the *disclosure* threat, i.e., the danger of eavesdropping on the exchanges between managed agents and a management station.	This protection is orthogonal to the ones offered by MarketNet. It can be achieved through encryption of the message (e.g., through the CBC-DES Symmetric Encryption Protocol as suggested in SNMPv3).
Partial immunity against the *message stream modification* threat. Protects against: • excessively delayed messaged • replayed messages but not against re-ordered messages (which could also occur in normal conditions).	*Equivalent protection*. Based on the timestamps and the unique identifiers carried by payments.
No protection against *Denial of Service (DOS)* attacks.	*Controllable*. DOS is limited by the budget available to the attacker and access prices. Both the budget dissipation and the prices are controlled by the GRM.

C. *Performance evaluation of SNMPvM*

Protection technologies using cryptography incur overheads due to crypto-operations. MarketNet is no exception, but can perform several operations off-line with respect to the actual transactions. This section analyzes the cost of protecting SNMP through MarketNet. Specifically it quantifies the performance of the MSK components depicted in Fig. 4 and measures the total transaction times for a basic test that consists of two SNMP applications – *snmpget* and *snmpwalk* – in which the client can access the server through a local area network. No currency-related operations are performed in advance. Transaction overheads for the following four variations of the basic scenario are also provided: (i) significant roundtrip (*rt*) delays, (ii) caching of foreign currency, (iii) advance purchase of foreign currency, and (iv) providing service prior to depositing the payment with its local bank.

The testbed used for measuring the performance of SNMPvM consisted of two 400 MHz Pentium II PCs running Linux 2.0.36, each implementing a separate currency domain with its own bank server, currency, and currency dissipation policy. The implementation and performance measurements of SNMPv3 and SNMPvM are based on the SNMP distribution ucd-snmp-3.6.1 provided by the University of California at Davis [8].

In the prototype implementation of MarketNet, currency is encrypted using 512-bit public and private keys. Encryption with asymmetric keys forms the basis for undeniable accountability in the use of resources. Accountability does not come for free. Compared to symmetric shared secret key encryption it incurs a much higher computational cost. In the prototype implementation and the testbed described here, RSA [9] encryption routines using 512 bit keys and the DES-CBC algorithm, require approximately 3.8 msec for encryption, and 22 msec for decryption, for a relatively long structure representing currency. Similar results are reported in [10] for measurements conducted on an IBM RS/6000, model 43P-200, with a PowerPC 604e CPU at 200 MHz, running AIX 4.2, for a variety of key lengths. They report latencies of approximately 2 msec for encryption and 10 msec for decryption with a 512 bit key.

TABLE IV presents the latencies associated with currency withdrawal, exchange and depositing in MSK. Notice that in most network applications and services, caching of foreign currency by domain banks and/or advance withdrawal of currency by client applications in anticipation of their short-term needs, will eliminate withdrawal latencies. Notice further that servers that receive currency as payment for the services they provide do not need to deposit the currency before providing the service; instead they can perform local validation of the received payment. Local validation is typically referred to as *off-line* verification, whereas validation by the bank is referred to as *on-line* verification. Off-line systems eliminate deposit delay, but run the risk of eventually depositing currency that has already been used by the customer for some other purchase.

TABLE IV
Banking-related latencies in MSK

Transaction Type	Latency (msec)
Withdraw local currency	59.5
Withdraw foreign currency	125.9
Deposit currency	61.5

TABLE V examines the latency incurred by the *snmpget* and *snmpwalk* applications. *snmpget* fetches a single object from a management information base (MIB), while *snmpwalk* fetches all the objects of a MIB. The particular instance of the *snmpwalk* presented, fetched 590 objects. Line 1 presents the latency in SNMPv1. Notice that SNMPv1 does not offer security. It is used here as a reference latency for better understanding of the protection-related performance overheads of SNMPvM and SNMPv3. To understand the latencies presented in lines 2–7 it is instructive to decompose a transaction into its constituent components. As an example, consider line 2 for the *snmpget* application. From Fig. 4 the total delay for the transaction consists of the following components: (actual *snmpget* transaction) + (withdrawal of foreign currency) + (deposit of payment). The latencies for the individual components of the transaction in TABLE IV and in line 1 of TABLE V add up to the value in line 2 (i.e., (7+125.9+61.5) msec = 194.4 msec for *snmpget* and (554 + 590*125.9 + 590*61.5) msec = 111,120 msec for *snmpwalk*). The roundtrip (rt) delay in lines 3–7 is the average delay measured between cs.columbia.edu and cs.berkeley.edu—approximately 67 msec.

TABLE V
Case-study – *snmpget* and *snmpwalk*

	***snmpget* latency (msec)**	***snmpwalk* latency (msec)**
1. SNMPv1	7	554
2. SNMPvM	194.4	111,120
3. SNMPv1 w/ rt delay	74	40,084
4. SNMPvM w/ rt delay	328.4	150,650
5. SNMPvM with remote currency caching at local bank & rt delay	135.7	111,474
6. SNMPvM w/ remote currency caching at local bank, rt delay, and off-line verification of currency	155.5	88,169
7. SNMPvM w/ advance acquisition of currency, rt delay, and off-line verification of currency	96	53,064

Line 3 demonstrates the effect of roundtrip delays, that is, the roundtrip delay becomes the dominant factor eventually.

In MarketNet, local domain banks observe the spending patterns of their customers and acquire in advance foreign currency in anticipation of near future demand. The exchange of currencies is typically done in advance and off-line from the actual transactions. Line 5 demonstrates the importance of caching of remote currencies at the local bank.

In many applications it is sufficient for the service provider to verify *off-line* the validity of the received payment locally without depositing the currency. It is enough in such cases to guarantee – as MarketNet does – that in case of double-depositing of currency, the bank can provide undeniable proof that either the service provider double deposited the currency (directly or through some third entity), or the customer double-spent it. The savings in time resulting from off-line verification are demonstrated in line 6.

Repetition of specific tasks and request for specific services constitutes the prevailing pattern of application operation. Such patterns can be exploited to perform advance acquisition of the currency required by the majority of applications. Local storage of this currency that is passed to the applications at the time of their initiation, avoids the cost of withdrawing/exchanging currency. The savings in time resulting from both advance purchase of currency and off-line verification are demonstrated in line 7. Comparing lines 3, 6 and 7 demonstrates the feasibility of the MarketNet approach even in the non-optimized prototype implementation.

A fair judgment of the overheads of protecting SNMP through MarketNet, should involve comparison of the overheads presented in TABLE V against those in SNMPv3. At the time of writing, few implementations of SNMPv3 exist. Most of them implement only part of the functionality specified in [3] and primarily focus in interoperability rather than performance. The initial measurements conducted on the SNMPv3 prototype implementation provided by the University of California at Davis (UCD) [8] with the agent running at the UCD site (on port 161 in ucd-snmp.ucdavis.edu) and the applications running locally at cs.columbia.edu, delivered the following average measured latencies:

- roundtrip delay: *71.3 msec*
- *snmpget* latency using SNMPv3 with authentication through MD5 but without encryption for privacy: *469.2 msec*
- *snmpget* latency using SNMPv.1: *215.1 msec*

IV. RELATED WORK

MarketNet is a novel approach to network security providing the ability to quantify and tune exposure to attacks, account for the use of resources, correlate attack detection and coordinate response, identify and isolate attack sources, and deploy continuous dynamic access control. Typical network security technologies only focus on some of these security problems. Some of the security concepts in MarketNet, such as the ability to quantify and tune exposure to attacks are novel contributions and, to the best of our knowledge, nonexistent in other technologies. Our presentation of related work in this section is a very brief summary of similarities and differences between MarketNet and existing categories of network protection technologies including Access Control Lists (ACL) and capabilities-based systems, intrusion detection mechanisms, authentication infrastructures, firewalls, and electronic payment infrastructures.

ACL and capabilities-based systems [11-13] both try to restrict resource access to a specific set of authorized users. In ACL it is necessary to identify the source of every request. Coupled with authentication mechanisms, this can provide strict resource control. Access is granted when the entity requesting access is among the list of permissible users. The decision is binary – an entity either has or does not have the right to access a resource – and the amount of access or the number of times an authorized entity accesses a resource is unrestricted. Access permission is static, that is, it is hard to continuously adjust the set of users that have access based on the operating conditions of the object or on security considerations. ACL do not scale since an entry is required for every entity that has access to the object being secured. Capabilities-based systems provide security that is similar to that in ACL. MarketNet is more powerful and more flexible than either ACL or capability-based systems. It provides a continuous range of access capabilities that depends on the budget of the client and the price to access a resource. The necessary condition to access a resource in MarketNet is the possession of the required budget, avoiding the scaling problem of ACL. None of the security features is given up: the source of the access can be traced back through the currency used. And, similar to an ACL-based access system, an object can have its own currency, which is restricted to some participants. Finally, access to resources in MarketNet is dynamically controlled through prices. Rising prices enable access only to priority clients holding the necessary budget.

MarketNet uses existing intrusion detection mechanisms [14] (and is not necessarily providing new ones). However useful intrusion detection mechanisms may be, there exist two problems that currently limit their application. First, they identify the fact that a system has been intruded after the intrusion – and potential damage – has taken place. Second, competent intruders typically manage to hide their identities and the intrusion cannot be traced back to the responsible entity. MarketNet can improve intrusion detection systems by allowing resource managers to impose limits on exposure to attacks, and by imposing traceable accountability.

In Kerberos [15, 16], one of the most prominent authentication systems today, tickets authenticate the identities of the entities involved in a transaction, and provide unlimited access to a resource during the validity of the ticket. MarketNet differs in a few fundamental ways from Kerberos. First, MarketNet imposes dynamically adjustable limits on the use of resources. The currency that clients hold has value associated with it; access to resources is priced and prices can be tuned in response to congestion or attacks. Limiting the *amount* of access to resources, i.e., quantifying access, is a fundamental departure from Kerberos. MarketNet uses *value* to quantify and adjust exposure to attacks, and even prevent many attacks by restricting the budget it allocates to potential attackers below the levels that would be needed to perform certain attacks. The second fundamental difference from Kerberos lies in the trust delegation model, the bookkeeping and use of resource access information, and the ahead-of-time potential acquisition of access rights by clients. Collectively, these mechanisms provide a more flexible, powerful, and potentially more efficient security model than Kerberos. In MarketNet, delegation of access rights is performed in a distributed fashion. Trading of access rights is entirely under the control of the respective domains and involves only the two domains exchanging currencies without the inter-mediation of any other external authority. Finally, MarketNet introduces undeniable liability in the use of resources. The association of specific payments with specific requests that could only have been generated by specific entities can be indisputably proven to a third party.

Firewalls [17] are currently an indispensable protection of attack-prone Internet sites against malicious entities residing outside the firewall. Firewalls typically protect through authenticating the source of requests and by observing the contents of incoming and outgoing packets. In most practical cases firewalls become performance bottlenecks and competent intruders still manage to circumvent them. Furthermore, firewalls cannot account for insecurities introduced by new programs. Lastly, malicious users residing inside the network protected by the firewall may not have to go through the firewall to access local resources. MarketNet overcomes these limitations since it emphasizes only on whether one has the appropriate currency to pay for the price of accessing a resource, independent of “how” one manages to get to the resource. The distinction between internal and external users is eliminated; malicious attackers that have managed to get through the firewall as well as new programs that introduce security holes will still need currency – maybe not available to them – to access the resources and cause damage.

Electronic payment systems [18, 19] have developed many robust protocols [18-28] over the last decade for the secure creation and dissipation of electronic money over insecure networks. They emphasized trading goods (such as books, clothes, etc.) safely over the Internet. MarketNet has greatly benefited from key design ideas in these. The protocol suite developed for MarketNet focus primarily on efficiency.

V. Conclusions

MarketNet develops novel technologies that shift power from attackers to defenders, giving the defenders control over the exposure to attacks, detection, and accountability of attackers. MarketNet uses market-based techniques to regulate access to resources. MarketNet technologies enable unified monitoring and correlated analysis of resource access to detect intrusion attacks, isolate the sources of attacks and respond quickly to control their damages. In summary, some of the key ideas in MarketNet are the following.

- Access to a resource must be paid-for with currency issued by its domain.
- Domains dynamically control and fine-tune the power of attackers by limiting the budgets allocated to them, and control the exposure of resources by setting their prices (for quantifiable access control).
- Domains monitor currency flows and use uniform, resource-independent statistical algorithms to correlate and detect access anomalies indicating potential attacks; this information is shared among domains to establish coordinated response to attacks.
- Currency is marked with unique non-forgeable identifiers that permit domains to establish verifiable liability for access to their resources, and identify and isolate sources of attacks.

MarketNet mechanisms unify and kernelize global information systems protection by containing all protection logic in a small core of software components.

MarketNet technologies have been implemented and successfully applied to the protection of a variety of Internet services.

Acknowledgement

The authors would like to thank Henning Schulzrinne, Sal Stolfo, Doron Avital and Gur Huberman for extremely helpful discussions and suggestions on the material presented in this paper.

References

[1] A. Dailianas, Y. Yemini, and D. Florissi, “The MarketNet Security Kernel (MSK) Design and Implementation,” Computer Science Department, Columbia University, New York, Technical Report cucs-032-99, November 1999.

[2] S. Kent, “Security Architecture for the Internet Protocol,” Request for Comments 2401, November 1998.

[3] U. Blumenthal and B. Wijnen, “User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3),” Request for Comments 2574, April 1999.

[4] J. Case, R. Mundy, D. Partain, and B. Stewart, “Introduction to Version 3 of the Internet-standard Network Management Framework,” Request for Comments 2570, April 1999.

[5] G. Waters, “User-based Security Model for SNMPv2,” Request for Comments 1910, February 1996.

[6] K. McCloghrie, “An Administrative Infrastructure for SNMPv2,” Request for Comments 1909, February 1996.

[7] W. Stallings, *SNMP, SNMPv2, and CMIP*. Reading, MA: Addison-Wesley, 1993.

[8] “The University of California at Davis SNMP Home Page,” *ucd-snmp.ucdavis.edu*.

[9] “RSA Data Security,” *http://www.rsa.com*.

[10] V. P. George Apostolopoulos, Debanjan Saha, “Transport layer security: How much does it really cost?,” presented at Conference on Computer Communications (IEEE Infocom), New York, 1999.

[11] “The KeyKOS Operating System,” *http://www.cis.upenn.edu/~KeyKOS/*.

[12] “EROS: The Extremely Reliable Operating System,” *http://www.eros-os.org/*.

[13] H. M. Levy, *Capability-Based Computer Systems*. Bedford, Mass: Digital Press, 1984.

[14] E. G. Amoroso, *Intrusion Detection : An Introduction to Internet Surveillance, Correlation, Traps, Trace Back, and Response*, 1st edition ed, 1999.

[15] C. Kaufman, R. Perlman, and M. Speciner, *Network Security - Private Communication in a Public World*, 1995.

[16] J. Kohl and C. Neuman, “The Kerberos Network Authentication Service (V5),” RFC 1510, September 1993.

[17] W. R. C. a. S. M. Bellovin, *Firewalls and Internet Security*: Addison-Wesley Professional Computing Series, 1994.

[18] D. O'Mahony, M. Peirce, and H. Tewari, *Electronic Payment Systems*: Artech House, 1997.

[19] B. Schneier, *Applied Cryptography : Protocols, Algorithms, and Source Code in C*, 2nd edition ed: John Wiley & Sons, 1995.

[20] “Digicash,” *http://www.digicash.com/*.

[21] “CyberCash,” *http://www.cybercash.com*.

[22] “Netbill,” *http://www.netbill.com/netbill/protocol.html*.

[23] “Mondex,” *http://www.mondex.com/*.

[24] “Millicent,” *http://www.millicent.com/*.

[25] “Secure Electronic Transactions (SET) Protocol,” *http://www.setco.org/set_specifications.html*.

[26] M. Bellare, J. Garray, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner, “iKP - A family of secure electronic payment protocols,” presented at First USENIX Workshop on Electronic Commerce, New York, 1995.

[27] T. Okamoto and K. Ohta, “Universal Electronic Cash,” in *Advances in Cryptology - CRYPTO '91*, vol. 576, *Lecture Notes in Computer Science - International Organization for Cryptologic Research*, J. Feigenbaum, Ed. Santa Barbara, California, 1991.

[28] T. Pedersen, “Electronic Payment of Small Amount,” presented at Cambridge Workshop on Security Protocols, 1996.