

AFRL-IF-RS-TR-2002-200
Final Technical Report
August 2002



MARKETNET: A SURVIVABLE, MARKET-BASED ARCHITECTURE FOR LARGE-SCALE INFORMATION SYSTEMS

Columbia University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. F249

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

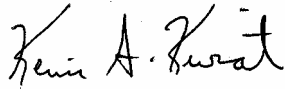
The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-200 has been reviewed and is approved for publication

APPROVED:



KEVIN A KWIAT
Project Engineer

FOR THE DIRECTOR:



WARREN H. DEBANY, Jr., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Aug 02	3. REPORT TYPE AND DATES COVERED Final Jul 97 – Dec 01	
4. TITLE AND SUBTITLE MARKETNET: A SURVIVABLE MARKET-BASED ARCHITECTURE FOR LARGE-SCALE INFORMATION SYSTEMS			5. FUNDING NUMBERS C - F30602-97-1-0252 PE - 62301E PR - F249 TA - 40 WU - 08	
6. AUTHOR(S) Yechiam Yemini, Apostolos Dailianas and Danilo Florissi				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Columbia University Distributed Computing & Communications Lab 450 Computer Science Building New York City, NY 10027			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency AFRL/IFGA 3701 North Fairfax Drive 525 Brooks Rd Arlington, VA 22203-1714 Rome, NY 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2002-200	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Kevin A. Kwiat, IFGA, 315-330-1692, kwiatk@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) The MarketNet project has developed novel information systems protection mechanisms based on market-based paradigms. These mechanisms seek to ensure the systematic, quantifiable, and predictable survivability of large-scale information systems. Specifically, MarketNet has pursued the following objectives: a) Control access to protected resources and domains and establish quantifiable and tunable limits on the power of attackers to access or damage critical information systems resources; b) Establish full accountability among separately administered and mutually distrustful domains, and enable rapid tracing and isolation of attack sources; c) Provide resource-independent instrumentation to monitor resource access, detect intrusion attacks automatically, identify their sources, rapidly isolate attack sources, and deny access, and d) Provide quantifiable protection against loss of critical resources due to attacks or failures. The project has successfully accomplished the design and development of substantial new technologies for protecting systems and applications, including the software implementation of the core MarketNet mechanisms and protection of several network services.				
14. SUBJECT TERMS Market-based System Protection Mechanism, Distributed System Survivability				15. NUMBER OF PAGES 27
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Abstract

The MarketNet project has developed novel information systems protection mechanisms based on market-based paradigms. These mechanisms seek to ensure the systematic, quantifiable, and predictable survivability of large-scale information systems. Specifically, MarketNet has pursued the following objectives: (A) Quantify and control access to protected resources and domains. (B) Establish tunable limits on the power of attackers to access or damage critical information systems resources. (C) Establish full accountability among separately administered and mutually distrustful domains, and enable rapid tracing and isolation of attack sources. (D) Provide resource-independent instrumentation to monitor resource access, detect intrusion attacks automatically, identify their sources, rapidly isolate attack sources, and deny access. (E) Provide quantifiable protection against loss of critical resources due to attacks or failures.

The project has successfully accomplished the design and development of substantial new technologies for protecting systems and applications, including the software implementation of the core MarketNet mechanisms and protection of several network services. Specifically:

- The MarketNet system, the first market-based protection system;
- The MarketNet-based protection of network services such as SNMP and the Java Virtual Machine;
- Mechanisms for quantification and tuning of exposure to attacks;
- Market-based mechanisms for intrusion and undesirable access detection; and
- Application of Market-based mechanisms for the protection of distributed systems.

Table of Contents

EXECUTIVE SUMMARY	1
1. TASK OBJECTIVES.....	3
1.1 THE NEED FOR NEW MECHANISMS FOR SYSTEM PROTECTION	3
1.2 RESEARCH OBJECTIVES	3
2. TECHNICAL APPROACH.....	4
2.1. MARKETNET FEATURES	4
2.2. MARKETNET ARCHITECTURE.....	4
2.3. BASIC OPERATIONAL SCENARIOS.....	6
2.3.1. <i>Adding a new client to the set of existing clients</i>	6
2.3.2. <i>Adding a new service to the set of protected services</i>	7
2.3.3. <i>Client accesses a service</i>	8
2.4. APPLYING MARKETNET MECHANISMS.....	9
2.5. EXAMPLES OF PROTECTIONS PROVIDED BY MARKETNET	10
3. GENERAL METHODOLOGY	10
4. TECHNICAL RESULTS.....	11
5. IMPORTANT FINDINGS AND CONCLUSIONS.....	11
6. SIGNIFICANT DEVELOPMENT	11
7. SPECIAL COMMENTS.....	12
8. IMPLICATION FOR FURTHER RESEARCH.....	12
APPENDIX A: PROTECTING DISTRIBUTED SYSTEMS	13
A.1. PROTECTING GENERAL SERVERS.....	13
A.1.1. <i>Providing Accountability in MarketNet</i>	14
A.1.2. <i>Mechanisms for Limiting Access and Attack Power in MarketNet</i>	14
A.1.3. <i>Preventing Attackers from Bypassing the MarketNet Protections</i>	17
A.2. PROTECTING REPLICATED DISTRIBUTED SERVICES	18
A.2.1. <i>Typical Replication Scenario</i>	19
A.2.2. <i>Protecting Against Server Compromises</i>	19
A.2.3. <i>Quantifying and Dynamically Limiting the Voting Power of Attackers</i>	20
A.2.4. <i>Unique Properties of the MarketNet Protection For Replicated Services</i>	21

Table of Figures

FIGURE 1: USE OF CURRENCY FOR ACCESS CONTROL IN MARKETNET	5
FIGURE 2: ADDING A NEW CLIENT TO THE SET OF EXISTING CLIENTS	6
FIGURE 3: ADDING A NEW SERVICE TO THE SET OF PROTECTED SERVICES	7
FIGURE 4: CLIENT ACCESSES A SERVICE.....	8
FIGURE 5: PROTECTING A CLIENT-SERVER APPLICATION THROUGH MARKETNET	14
FIGURE 6: DYNAMIC PRICING POLICIES.....	15
FIGURE 7: LEAKY-BUCKET BUDGET EXPENDITURE CONTROL MECHANISM	16
FIGURE 8: PROTECTING ACCESS TO A RESOURCE	17
FIGURE 9: NETFILTER-BASED IMPLEMENTATION OF MARKETNET PROTECTION	18
FIGURE 10: TYPICAL VOTING SCENARIO	19

EXECUTIVE SUMMARY

This report summarizes the research conducted under the DARPA research project “MarketNet: a Survivable Market-based Architecture for Large-scale Information Systems”. This executive summary focuses on the main research goals and results.

The challenge addressed by this project is how to accomplish quantifiable, predictable survivability of large-scale information systems under loss or rapid dynamic changes in availability of resources. The term resource is used in its broadest form to mean a physical resource such as processor, memory, storage, bandwidth or sensors, or a higher level service resource such as a file server, database server, name server, web server, or a particular data or application software. Loosely speaking, survivability means that client applications requiring use of these resources can adapt to reduced resources or use alternate resources to execute, according to their intrinsic priority. It means that resource managers can reallocate resources to best reflect the needs and priorities of clients. It also means that once an attack has been identified, the system can rapidly deploy protection of its healthy parts against further loss and damages.

MarketNet establishes mechanisms in networked information systems to support economic-based decentralized resource management. Resources and their consumer clients are organized in currency domains, each with its own currency. Consumers of resources or services must purchase their access using currency acceptable by the domain owning these resources. Each domain fully controls the budget allocated to a given client, whether an internal client or an external distrusted domain. Providers dynamically adjust prices of the resources offered. Resources are dynamically replicated. Replication is governed by economic optimization of the expected benefits to the provider. Consumer processes select among replicas to optimize their cost; thus accomplishing optimized load balancing.

The MarketNet project has developed novel information systems protection mechanisms based on market-based paradigms. These mechanisms seek to ensure the systematic, quantifiable, and predictable survivability of large-scale information systems. Specifically, MarketNet has pursued the following objectives: (A) Quantify and control access to protected resources and domains. (B) Establish tunable limits on the power of attackers to access or damage critical information systems resources. (C) Establish full accountability among separately administered and mutually distrustful domains, and enable rapid tracing and isolation of attack sources. (D) Provide resource-independent instrumentation to monitor resource access, detect intrusion attacks automatically, identify their sources, rapidly isolate attack sources, and deny access. (E) Provide quantifiable protection against loss of critical resources due to attacks or failures.

The MarketNet project has successfully accomplished the design and development of substantial new technologies for protecting systems and applications. Specifically, the project developed the following technologies.

1. The MarketNet system, the first market-based system protection mechanism.

MarketNet is a set of technologies for the survivability and protection of information systems. Resources and services in MarketNet are organized in currency domains. Resources are instrumented to charge for their use. To access a resource, a client must pay in currency accepted by the resource owner. Resource owners can control the power of attackers by limiting the budgets available to them, and by setting the prices to access the resources they own, 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. MarketNet mechanisms unify and kernelize global information systems protection by containing all protection logic in a small core of software components.

2. The MarketNet-based protection of network services.

MarketNet can protect any existing and future client-server application without modifying its operation or implementation. In order to demonstrate the feasibility of this approach to protect real

systems, the project pursued the protection of SNMP Version 1 (SNMPv1) and the Java Virtual Machine (JVM). The insecure version of SNMP (SNMPv1) has been extended with MarketNet mechanisms to limit access and changes to MIB variables. The project demonstrated that the protection of SNMP through MarketNet provides equal or stronger levels of protection compared to the secure version of SNMP (SNMPv3), incurring minimal performance overheads. The JVM was extended with MarketNet mechanisms for controlling prices to access resources and for restricting budget expenditure on a per-thread basis in order to limit processor and memory consumption. Both protection systems were developed at fraction of the development time that would be required to develop specialized protection mechanisms in the protocol (such as the ones in SNMPv3) or in the Java engine.

3. Mechanisms for quantification and tuning of exposure to attacks.

Resource owners in MarketNet can determine and adjust the exposure of resources to attacks, and the power of users to attack resources. A client accessing a resource or service in a typical transaction pays the resource manager with the appropriate amount and type of currency. Currency domains control the allocation of budget to all entities according to their currency dissipation policies. Resource managers control and dynamically adjust the prices of their resources, and impose restrictions on the way customers spend their budget to access resources. Based on these mechanisms, the project developed mechanisms to limit the power of users to access resources, and to quantify and tune the potential effects of attacks.

4. Market-based mechanisms for intrusion and undesirable access detection.

Analysis of MarketNet currency flows can reveal access anomalies and detect intrusions and lead to important new technologies for intrusion detection. Access monitoring and quantification of the attack damage can be expressed in a uniform, resource and service-independent access metric: currency value transferred as a result of the access. Unlike current intrusion detection techniques that deal with intentions and detect either anomalous behaviors or specific attack patterns, MarketNet detects accesses that cause excessive financial damages to the resources or services and are therefore undesirable. In addition, the detection process can influence user behavior through price manipulations and force attackers to reveal their hidden actions and intentions. Finally, access-source information in MarketNet enables detection of illegal accesses, and the analysis of individual or group behaviors to identify access anomalies. The project exploited such MarketNet-based mechanisms in several real attack scenarios.

5. Application of Market-based mechanisms for the protection of distributed systems.

The MarketNet principles and mechanisms are not restricted to simple client-server protections. A very promising novel area of application is to the protection of replicated distributed services, where critical services are replicated to tolerate failures of a minority of nodes through a voting scheme that determines the majority result of the service. Replication in itself is considered insufficient to protect against malicious attacks. The initial analysis indicates that replication combined with MarketNet protection can be a particularly effective technique against attacks to critical services.

Topics 1-4 are summarized in this report. More details can be found in the accompanying documents published during the time of the original contract. Topic 5 is described in Appendix A for completeness because it reflects work done more recently, during the latest extension to the original contract.

1. TASK OBJECTIVES

1.1 THE NEED FOR NEW MECHANISMS FOR SYSTEM PROTECTION

Protecting access to networked information systems and services remains an elusive challenge of ever-growing importance and complexity. Existing systems exhibit an imbalance of power between attackers and defenders. The power to attack systems is controlled by the attackers. These systems and services are increasingly exposed to attacks, resulting in significant disruption of services and damages. On the other hand, defenders have no way to systematically control access to their resources, and assess their exposure to attacks.

Current network systems offer several opportunities to attackers that are not fully addressed by current protection mechanisms. First, they are organized based on trust among components. An attacker can thus compromise one component to take advantage of the trust extended to it by other components and gain access to their services. Second, attackers have unlimited opportunity to attempt access to resources, evade detection and hide their identity. Third, there is lack of unifying security architecture. Protection is too often left at the mercy of human errors in ad-hoc designs and configurations.

Given these trends, there is a need for a set of unified, symmetric protection mechanisms. To this end, several technical challenges must first be addressed.

1. How to enable unified, transparent, quantified management and control of trust among distrustful entities?
2. How to shift power from attackers to resource administrators and enable the latter to strictly control the amount and level of access provided to their resources, assess and bound the exposure of their resources to attacks, detect and prevent attacks and undesirable access behaviors in real time, and establish the identity of the attack source in a manner that permits strict accountability?
3. How to provide unified, transparent protection that can sustain the large number and variety of technologies and products incorporated in networked systems and remain resilient to changes?
4. How to develop protection mechanisms that can attain the above goals while exhibiting low overheads and high degree of security of the protection infrastructure?

The MarketNet project seeks to address these challenges.

1.2 RESEARCH OBJECTIVES

The primary goal of the MarketNet project was to develop new network protection technologies, apply these technologies to protect existing systems, and study their relationship with existing proposals.

More specifically, the research focused on the following goals.

1. Developing new market-based protection technologies.
2. Apply the market-based protection technologies to the protection of existing protocols.
3. Apply the market-based protection technologies to the protection of existing systems.
4. Develop mechanisms that can quantify and tune exposure to attacks.
5. Develop market-based mechanisms for intrusion and undesirable access detection.
6. Exporting the resulting technologies to DOD, to industry, and to the research community.

Notice that these goals substantially refine and expand the original goals stated in the proposal and subsequent contract. This refinement and extensions reflect the improved understanding and results gained through the project.

2. TECHNICAL APPROACH

The MarketNet architecture and protection mechanisms have evolved over the years. This section provides an overview of the latest developments and simplifications of the MarketNet architecture and mechanisms and how they address the challenges outlined in Section 1.1, and provides some brief examples of protection for well-known attacks. More details, including the general MarketNet architecture, can be found in attached documents listed in Section 4.

2.1. MARKETNET FEATURES

MarketNet has developed novel protection paradigms and infrastructure that reverse the power imbalance between attackers and defenders in favor of the defenders and allow entities to control and quantify the allocation of trust. In particular MarketNet:

1. Enables unified, transparent, quantified management and control of trust among distrustful entities
2. Shifts power from attackers to resource administrators; in particular, it enables administrators to (a) strictly control the amount and level of access provided to their resources, (b) assess and bound the exposure of their resources to attacks, (c) detect attacks in real time, and (d) establish the identity of the attack source in a manner that permits strict accountability
3. Provides unified protection mechanisms that can sustain the large number and variety of technologies and products incorporated in networked systems and remain resilient to changes
4. Involves low overheads and high degree of security of the protection infrastructures

2.2. MARKETNET ARCHITECTURE

MarketNet organizes resources and services in currency or administrative domains and instruments resources to use currency for access control and monitoring. Clients wishing to access a target resource must pay the access price with the currency of the target domain. Figure 1 depicts the overall conceptual organization and interaction of the entities in the MarketNet architecture.

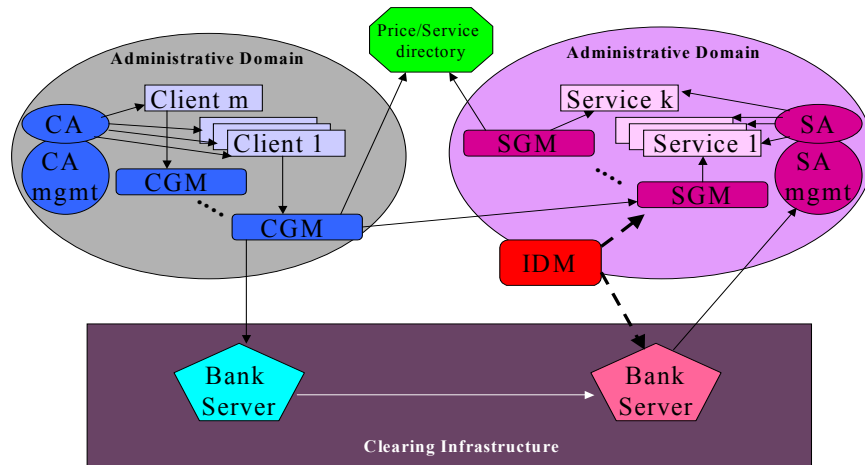


Figure 1: Use of currency for access control in MarketNet

The **clearing infrastructure** maintains accounts similar to a bank, and provides similar functionality, i.e., provides withdrawal, deposit, transfer, and exchange capabilities. It further logs information that can be used for liability and intrusion detection purposes. The clearing infrastructure is only partially trusted, e.g., it is trusted to keep logs, but not to control the allocation of currency. The service administrators (SAs) guarantee that banks do not misbehave by authorizing allocation of budget to requesting entities and later (at the time of access to the domain resources) validating this authorization through the service gating modules (SGM). Whenever a bank transfers currency to another bank or to a user, it creates and records an association that can later be used to prove liability of the recipient for the use of currency.

Service Administrators (SAs) protect one or more resources. A SA assigns a resource to an access class. For each access class an SA determines the amount of access that individual client administrators (CAs) or collections of CAs can have at any point in time. SAs do not trust CAs, but they can hold them liable for accesses to resources they control. SAs establish prices for accessing resources they manage; accept and verify payment; verify the association of payment with the access it is financing; and can restrict the way clients spend their budget to limit the way clients can access the resources they manage, regardless of the budget clients hold.

A **service or resource** is controlled by one SA only and belongs to one of a few access classes (e.g., public restricted, confidential).

Service Gateway Modules (SGMs) act as gates through which all accesses to resources/services are performed. SGMs analyze requests and determine whether the payment to perform the request carries the appropriate amount and type of access rights.

Client Administrators (CAs) control one or more clients. A CA assigns a user to an access class. The access class reflects the level of trust to individual users. The CA determines the access budget allocated to its users. A CA assumes liability for the actions of the users it controls. A CA that is held accountable for a particular access, uses the information logged in the associated bank to establish the identity of the user (among the ones it controls) that is responsible for the specific access.

A **user** can be controlled by more than one CAs, but for every one of its accesses there is a single CA that acts as the liable entity. A user belongs to a particular access classes (e.g., public restricted, confidential) where it is assigned by its CA.

A **Client Gateway Module (CGM)** acts as gate through which a user accesses resources. The CGM finds the amount and type of access rights required to perform a transaction, contacts the clearing entity to get them, and performs the transaction on behalf of the user.

Directory Services advertise the amount and type of access rights required to access a service, along with the clearing entities that trade these access rights.

MarketNet instruments resources and services with uniform monitoring of currency flows that capture access behaviors, regardless of the specific operational details of the clients or resources. The **Intrusion Detection Monitor (IDM)** monitors currency flows through bank servers, service gateway modules, and client gateway modules. It analyzes the spending patterns of customers and the revenue generation patterns of resources through statistical algorithms to detect anomalous behaviors that indicate potential attacks. For example, the IDM monitors the local bank server and CGM of a client to determine a pattern of payments from the client to a set of resources and evaluates the distance of this pattern from predetermined patterns of payments that correspond to attacks. A distance that is smaller than a threshold alludes to a potential attack. Similarly, the IDM monitors the SGM or the local bank of the monitored resource and evaluates the proximity between patterns of revenues generated by resources and predetermined attack patterns.

2.3. BASIC OPERATIONAL SCENARIOS

This section outlines the interaction of entities in the MarketNet architecture through the demonstration of the steps involved in three fundamental actions in the system: enabling a client to transparently access MarketNet protected resources; enabling a service to be transparently protected through MarketNet; and protecting the access of a client to a service.

2.3.1. ADDING A NEW CLIENT TO THE SET OF EXISTING CLIENTS

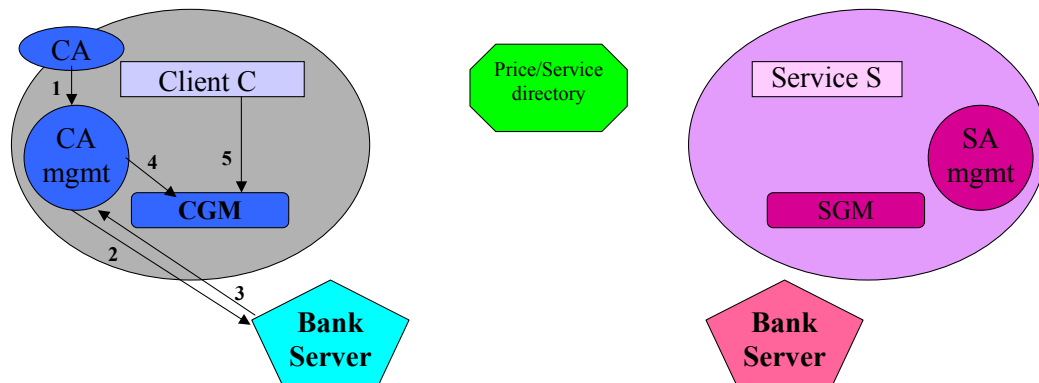


Figure 2: Adding A New Client To The Set Of Existing Clients

Adding a new client involves the following steps:

1. CA assigns the new client to (a) a specific access class, and to (b) a specific budget replenishment policy.
2. CA mgmt contacts the clearing entity and creates an account for the client. The password for the account is the client's public key.
3. The clearing module replies with a new account identifier.
4. CA mgmt instantiates a CGM and assigns it to the client. The new CGM is configured with (a) the access class of the client, (b) client account information, (c) the identifier of the CA liable for transactions of the client, (d) price/services directory information.
5. The client passes its private key to the CGM. Requests by the CGM to the clearing entity encrypted with this key, are stored by the clearing entity as proof of the client's liability for specific currency.

2.3.2. ADDING A NEW SERVICE TO THE SET OF PROTECTED SERVICES

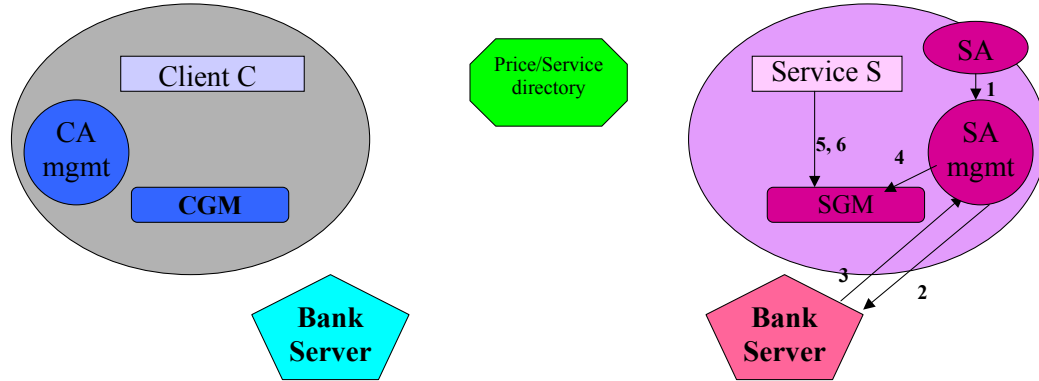


Figure 3: Adding a New Service To The Set Of Protected Services

Adding a new service involves the following steps:

1. SA assigns the new service to a specific access class. If the class has already been assigned to a budget dissipation policy, then the service inherits that policy as part of the access. If not, the SA creates a new budget dissipation policy and assigns it to the class, or assigns the class to one of the existing policies. Note that the access class and the budget replenishment policy are assigned independently to a particular client. In contrast the budget dissipation policy is fixed for an access class where services are assigned – in other words, the access class determines the budget dissipation policy at the service side.
2. SA mgmt contacts the clearing entity and creates an account for the service. The password for the account is the service's public key.
3. The clearing module replies with a new account identifier.
4. SA mgmt instantiates a SGM and assigns it to the service. The new SGM is configured with (a) the access class of the service, (b) a generic pricing policy that charges a fixed amount per request, (c) service account information, (d) the SA responsible for budget dissipation and the respective policy, and (e) price/services directory information.
5. The service passes its private key to the SGM. Requests by the SGM to the clearing entity encrypted with this key, are stored by the clearing entity and used for detection of double-depositing.
6. The service owner optionally configures the price table at the SGM and provides the SGM with the module that parses requests corresponding to the entries in the SGM.

2.3.3. CLIENT ACCESSES A SERVICE

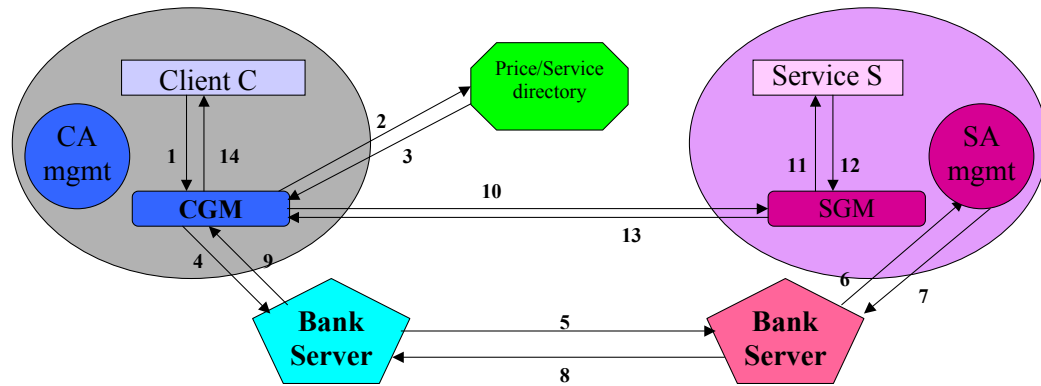


Figure 4: Client Accesses A Service

A typical access to a MarketNet protected service involves the following steps:

1. The client issues a request (original request)
2. The CGM sends the original request to the price directory.
3. The reply consists of:
 - (i) SA responsible; (ii) Access level; (iii) Price; and (iv) Clearing server location (e.g., IP address and port)
4. Request to withdraw money from client's account and exchange for target currency. Request includes:
 - (i) Currency domain; (ii) SA responsible; (iii) Access level; (iv) Amount; (v) Clearing server location; (vi) Entity liable for the request; and (vii) Account to withdraw the money from
5. Requests exchange of currency from clearing server location specified above. It includes all the info in the previous step, except the clearing server location and the account of the user. Notice that caching algorithms of the clearing entity may request an amount bigger than that to satisfy the current request
6. Clearing entity passes request to responsible SA management module.
7. SA mgmt module decides whether to honor the request or not. If yes, it logs the necessary information (e.g., may log liability information and adjust the remaining budget available to the particular liable entity), and then authorizes the transaction.
8. Clearing entity serving the SA logs the liability information and passes the reply back to the requesting clearing entity.
9. Clearing entity responsible for CGM logs liability information and passes currency to CGM.
10. CGM sends the request along with the payment to the server.
11. SGM validates request/payment and passes the original request to the server. Specifically it:
 - i. Extracts and validates payment (locally)
 - ii. Analyzes the request and discovers price for it
 - iii. Checks appropriate authorization level and sufficient budget in payment
 - iv. Logs request and payment locally
 - v. Periodically synchronizes (deposits) with clearing entity.

If the above checks (i, ii, and iii) succeed, the SGM passes the original request to the server, otherwise sends an error notification to the requester.
12. Server passes reply to SGM.

13. SGM relays the reply to CGM (need to consider change)
14. CGM does some processing related to the payment for the request, and passes the reply to the client.

Notice that most of the above steps involving acquisition of the appropriate currency by the client typically happen in advance, off-line from the transaction and do not add latency to the transactions.

2.4. APPLYING MARKETNET MECHANISMS

CA and SA act as MarketNet proxies on behalf of clients and resources respectively and make the operation and existence of MarketNet transparent. They protect without modifying the semantics, operation or implementation of resources or clients.

SAs determine and tune the power of clients to attack, and the exposure of resources to attacks. A client's power is determined by (a) the budget allocated to the CA, (b) the price to access a resource, and (c) restrictions imposed by the manager on how the client spends its budget. For example, a web server that charges 1 unit of currency for access to a page, knows that a client holding 100 units of currency can access the page 100 times. The exposure is similarly determined based on the collective budget of groups of CAs. Domains dynamically adjust one or both of prices and budgets, to control the attack power of users and the level of risk or exposure to attack of resources. For example, when a new method of attack is detected, the prices of the respective resources used in the attack can be immediately raised to prevent such attacks. A manager may respond to a Denial of Service (DoS) attack by raising the prices of the resources under attack and by enforcing restrictions on the use of available budget. Similarly, it may prevent DoS attacks by controlling the amount and rate of budget allocation to a CA or a collection of CAs.

MarketNet provides mechanisms to irrefutably determine both the identity of a user and its accesses, anytime after payment has been made from the user to the resource. This feature provides liability of clients for resource accesses and is achieved through two associations. The first association binds a particular bill, through its unique identifier, with an entity liable for its use. It is generated by the banks when the currency is distributed. The second association links a particular bill to the access it is financing. It is generated when the client pays for accessing a resource. At any point thereafter, the identity of the user responsible for a particular access can be irrefutably determined by first analyzing the second association to determine which electronic currency was used to access the resource and then the first association to identify the client.

Independent enforcement of distribution policies by SGMs guarantees that the effects of conquering any part of the banking infrastructure are localized. Even if the attacker conquers the bank for, say domain X, or even manages to get or produce an infinite amount of domain X currency, the attacker cannot damage domain Y, since domain Y's bank controls the exchange of currency. The bank of domain Y can simply refuse to exchange currency or severely limit the amount of Y currency the attacker can acquire.

The banks or the IDM audit the balances of accounts maintained by other banks, resource managers or users. They can scrutinize these balances to detect possible misuse of funds. For example, assume an attacker conquers a bank and misuses the balances of accounts in this bank by appropriating funds belonging to others to finance attacks. This causes inconsistencies between the accounting maintained by the entities whose currency was used by the attacker and the accounts in banks of the target resource managers and domains. These accounting inconsistencies are used to identify banks that have been compromised.

SA, CA and banks, collect monitoring information for flows of currency resulting from paying for service or resource accesses. Both the monitoring instrumentation and the monitoring information are uniform and independent of the monitored entity. The IDM can easily correlate such information coming from different sources. For example, it can correlate the information from processor usage and access to the password file, and subtract from that the expenditures of a group of non-attacking customers and examine if the resulting activity resembles known attack patterns. The IDM can use the assessment of risk exposure to define rational detection policies and thresholds. For example, access to a resource can be denied when the

risk of attack exceeds an acceptable level. Finally, the IDM can try to influence user behavior through the advertised prices that provide feedback to users, in order to help uncover attacks hidden in normal traffic. For example, to identify a group of attackers that hide their activity by performing a distributed DoS attack on a site offering stock quotes, the resource manager, instructed by the IDM, advertises an alternative site offering identical service at lower prices. The group of attackers is likely to insist in accessing the particular expensive site it was attacking, forcing attackers to reveal their intentions and identity.

2.5. EXAMPLES OF PROTECTIONS PROVIDED BY MARKETNET

MarketNet defends against a wide variety of attacks including DoS, Trojan horses, identity stealing, identity masquerading, runtime stack overflow, worms attacks, and others. This section outlines examples of such protections.

MarketNet alleviates or prevents DoS attacks. In a DoS attack, the attacker creates large loads of activities that saturate a given resource and prevent legitimate access. In the “backslash attack” against *Apache* web servers, the attacker submits requests with URLs containing many backslashes. The server devotes most of its resources in processing these requests and eventually becomes unable to process other requests. The attacker needs to finance the excessive access to resources and soon depletes its budget. The SA may further rise the prices seen by the attacker to deplete the attack budget faster. Alternatively, it may restrict the rate at which the attacker can spend its budget, effectively restricting the load the attacker is imposing on the service. The IDM monitoring the resource will see unusually high revenue generated by the resource, and may block further requests by the attacker, if the potential revenue loss caused by the abnormal behavior exceeds some tolerance level.

MarketNet also alleviates or prevents “worm” attacks. In the recent “Love letter” worm attack, a malicious program (worm) was electronically mailed to users as an attachment file. Unsuspecting users clicked on the attachment, essentially authorizing the worm execution with the same privileges as the legal user of the account. As part of its execution, the worm replaced files, and mailed itself to all entries in the address book of the victim. In MarketNet, the worm will either have to finance its activities itself, or request the victim to do so. In the former case, the attacker leaves a trace linking the illegal activity back to him. In the later case, the victim can finance the execution, but impose restrictions on the allocation and usage of the budget. For example, restricting the rate of budget allocated to the worm can prevent it from performing expensive operations such as replacing or deleting a file. Or, the victim may finance the worm with currency that is valid only for the currency domain of resources such as processor and memory, but not for access to the file system that may reside on a different currency domain. In the worst case, the worm gets enough budget to succeed in requesting deletion of files. Deleting files will soon deplete its budget, limiting the effects of the attack. Further, the IDM may soon notice the unusual expenditure pattern and block further requests. Once the attack is known, IDMs monitoring different systems, may share this information and recommend increase of prices to delete or replace files, to prevent imminent attacks. Protection against the general case of “worm attack”, highlights management of trust among protection domains in MarketNet. Gaining access to a host does not affect other hosts by exploiting trust between them. Any propagation of the worm requires an appropriate new budget with the currency of the target domain. Thus, a successful attack in one domain does not compromise the security of other trusted domains.

3. GENERAL METHODOLOGY

In general the research pursued design, development, and experimentation with software systems implementing the technology challenges discussed.

4. TECHNICAL RESULTS

The technical results accomplished by the research are described in the accompanying documents.

- *"MarketNet: Use of Currency for Access Control in Large-scale Information Systems,"* A. Dailianas, Ph.D. Thesis, Computer Science Department, Columbia University, New York, July 2000.
- *"MarketNet: Market-based Protection of Network Systems and Services - an Application to SNMP Protection,"* A. Dailianas, Y. Yemini, D. Florissi and H. Huang. In Proceedings of IEEE Infocom, March 2000, Tel Aviv, Israel.
- *"MarketNet: Protecting Access to Information Systems Through Financial Market Controls,"* Y. Yemini, A. Dailianas, D. Florissi and G. Huberman. Decision Support Systems Journal, vol. 28/1-2, pp. 205-216, 2000.
- *"MarketNet: Market-based Protection of Information Systems,"* by Y. Yemini, A. Dailianas, D. Florissi, and G. Huberman. In Proceedings of ICE'98, First International Conference on Information and Computation Economics, Oct. 1998, Charleston, SC.
- *"MarketNet: Using Virtual Currency to Protect Information Systems,"* by Y. Yemini, A. Dailianas, and D. Florissi, 7th Delos Workshop on Electronic Commerce, Second European Conference on Research and Advanced Technology for Digital Libraries, Sept. 1998, Heraklion, Crete, Greece.
- *"Use of Currency for Access Control in Large-scale Information Systems,"* by Apostolos Dailianas, Ph.D. Thesis Proposal Department of Computer Science, Columbia University, Sept. 1998, New York.
- *"MarketNet: A Market-based Architecture for Survivable Large-scale Information Systems,"* by Y. Yemini, A. Dailianas, and D. Florissi. In Proceedings of Fourth ISSAT International Conference on Reliability and Quality in Design, Aug. 1998, Seattle, WA.
- *"Market Based Protection of Information Systems,"* by A. Dailianas. DARPA Graduate Student Work-shop, Aug. 1998, Arlington, VA.
- "Protecting General Servers and Replicated Distributed Servers with MarketNet," Technical report. Also included in Appendix A.

5. IMPORTANT FINDINGS AND CONCLUSIONS

See summary and accompanying documents for details.

6. SIGNIFICANT DEVELOPMENT

The project has accomplished development of several significant novel protection technologies. The key contributions are the following.

- The design and implementation of the MarketNet technologies.
- The application of MarketNet to protect the JVM and SNMP.
- The study of the application of MarketNet for the survivability of distributed systems.
- Submission of 5 patents (pending).
 - Using Electronic Security Value Units to Control Access to a Resource.
 - Unified Monitoring and Detection of Intrusion Attacks in an Electronic System.
 - Quantifying the Risk and Limiting Exposure to Attacks in an Electronic System.
 - Identification of an Attacker in an Electronic System.
 - A Banking Infrastructure for Generating and Managing Access Rights in an Electronic System.
- Complete MarketNet software prototype and software to protect the JVM and SNMPv1.
- Publication of 6 articles and 1 Ph.D. thesis.

7. SPECIAL COMMENTS

None.

8. IMPLICATION FOR FURTHER RESEARCH

Several research areas, explored by this project, show significant promise for further research.

1. Application of MarketNet to the survivability of large information systems. This work applied the MarketNet technologies to the protection of individual protocols and systems. The MarketNet architecture is designed for scalability and survivability. A natural follow-up to this work is to apply MarketNet to protect large operational distributed systems.
2. Study of application of MarketNet policies. MarketNet enables the quantification of exposure to attacks by controlling and dynamically adjusting prices, budgets, and replenishment policies. Further study is needed to identify how such policies should be applied in different scenarios to limit the effects of attacks.
3. Study of application of MarketNet-based intrusion detection. The study of application of MarketNet technologies for intrusion detection deserves further development.

APPENDIX A: PROTECTING DISTRIBUTED SYSTEMS

The MarketNet principles and mechanisms can transparently protect any service in networked and stand-alone electronic systems. The accompanying documents demonstrate how MarketNet is applied to the protection of diverse systems and services such as the simple network management protocol (SNMP) and the Java Virtual Machine (JVM). This appendix applies these ideas for the protections of general servers in large-scale distributed systems.

The MarketNet principles and mechanisms are not restricted to simple client-server protections. A very promising novel area of application is to the protection of replicated distributed services, where critical services are replicated to tolerate failures of a minority of nodes through a voting scheme that determines the majority result of the service. Replication in itself is considered insufficient to protect against malicious attacks. The initial analysis indicates that replication combined with MarketNet protection can be a particularly effective technique against attacks to critical services.

The rest of this appendix is structured as follows. Section A.1 presents the application of MarketNet to the protection of general servers. Section A.2 applies the MarketNet mechanisms and principles to the protection of replicated distributed servers.

A.1. PROTECTING GENERAL SERVERS

Resource owners in MarketNet can determine and adjust the exposure of resources to attacks, and the power of users to attack resources. A client accessing a resource or service in a typical transaction (depicted in Figure 5) pays the resource manager with the appropriate amount and type of currency. Service administrators (SAs) determine the amount of access that individual client administrators (CAs) or collections of CAs can have at any point in time. Furthermore, they control and dynamically adjust the prices of their resources, and impose restrictions on the way customers spend their budget to access resources. Based on these mechanisms, MarketNet develops limits on the power of users to access resources, quantifies the potential effects of attacks and demonstrates how these effects can be tuned. For a more extended coverage of the limits on the power of attackers, the reader should refer to the accompanying documents.

Service administrators control the budget allocation through the currency dissipation policy for the services they control. These policies are enforced by the bank server responsible for the dissipation of currency acceptable for access to the particular services. The currency dissipation policy can control several parameters of budget allocation, such as: (a) the total budget allocated to an administrative domain or to any collection of administrative domains; (b) the rate at which this budget can be renewed; (c) the total currency collectively allocated to all administrative domains; etc.

A service administrator has several mechanisms to achieve its protection objectives. The choices have implications on security, manageability, performance and scalability. The first mechanism is the creation of a new currency domain dedicated to the protection of a specific set of resources. Accessibility under a new currency and control of the entities the new currency is allocated to, emulates access restrictions offered by access control lists. The second mechanism is control of prices to access resources. Prices can be static or dynamic and may be different for different customers. Static prices are used to quantify the amount of access a specific budget allows. Dynamic prices can reflect the operating conditions and security considerations of the resource managers and provide feedback to influence customer behavior. Differentiated prices can selectively vary the access power of different users. The third mechanism available to service administrators, is control of the way customers use their budget to access resources, independent of the budget they possess. Control of budget usage can be achieved through a mechanism such as the well-known *leaky bucket*.

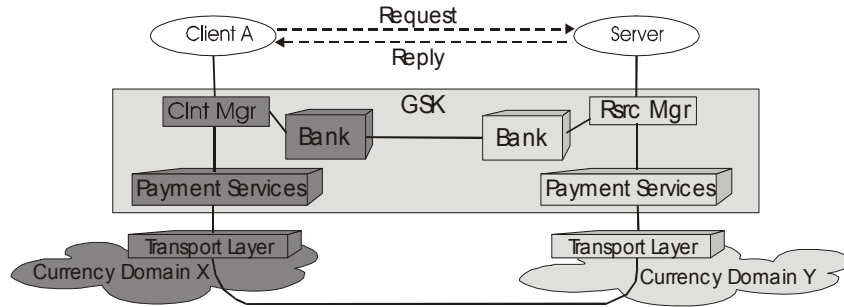


Figure 5: Protecting a client-server application through MarketNet

By controlling the prices and the budgets available to clients, service administrators quantify the amount of access rights a client has to their resource. They can change this amount dynamically by adjusting their prices. Setting their access prices sufficiently high, they strictly control dynamically who has access to what resources. Their ability to throttle attacks is reflected either by the lack of budget or the lack of expenditure privileges needed by attackers to perform an attack. Quantification of exposure assumes various forms depending on the particular resource and the particular kind of attack under consideration. Examples of such bounds have been developed in the accompanying documents; they include the duration of DoS attacks, the effects of DoS attacks, and access control restrictions in the case of critical resources. Application of these bounds to the Java Virtual Machine (JVM) has also been demonstrated.

The rest of this section provides an outline of the mechanisms available to resource managers for quantifying the access power of customers. For examples of tunable limits on the power of attackers and the damage they can cause to services the reader can refer to the accompanying documents.

A.1.1. PROVIDING ACCOUNTABILITY IN MARKETNET

MarketNet provides accountability in the use of resources. Accountability in practical terms means that any access to a resource can be traced back to the entity responsible for it. Furthermore, this association of an access to the responsible entity is undeniable and can be used as the basis for legal liability. Accountability is achieved through a collaboration of mechanisms. In particular, whenever an entity (e.g., a bank) transfers currency to another entity (e.g., another bank or a user), it creates and records an association that transfers the liability for the use of the particular bills to the recipient. Furthermore, currency for accessing a particular set of resources can only be authorized by the entity responsible for managing these resources. The payment protocols described in the accompanying documents guarantee that the validity of the currency received for accessing a resource can be verified by the manager of the resources. Finally, these protocols provide guarantees that currency cannot be duplicated, cannot be used for any purpose other than that intended by their legal owner (which amounts to preventing stealing of currency), and that there is an undeniable association between a particular bill and an entity that is trying to use it to access a particular resource.

A.1.2. MECHANISMS FOR LIMITING ACCESS AND ATTACK POWER IN MARKETNET

The power of users to access resources in MarketNet is restricted by (a) their available budget, (b) the prices to access resources, (c) restrictions on the way they spend their budget, imposed on them by service administrators, and (d) intrusion detection mechanisms that may decline their access requests to prevent undesirable usage of resources. An outline of these mechanisms follows.

Control of Budget Allocation

Allocation of budget for accessing a set of services is controlled and enforced by service administrators. For reasons of scalability, budget is allocated to client administrators, and not to individual clients. Budget allocation policies can enforce restrictions on several aspects of currency dissipation. Examples of such restriction, include the following:

- A_{tot} – maximum total amount allocated to external entities
- $A_{tot,i}$ – maximum amount allocated to a specific liable entity i
- R_{tot} – maximum total rate of currency outflow
- $R_{tot,i}$ – rate of currency outflow to a specific liable entity i

The accompanying documents develop several limits on the access and attack power of individual entities or collections of entities, based on the control of these parameters.

Price Control

Service administrators control the prices to access the resources they manage. The pricing policy can be used to protect resources. An example of such a pricing policy that protects access to the management information base (MIB) in the simple network management protocol (SNMP) is presented in the accompanying documents. MarketNet provides suggested pricing policies, along with a quantification of their protection characteristics that assist service administrators to assess and tune their exposure to attacks. We outline the general categories of such pricing policies.

Prices may be *fixed* or *dynamic*. Fixed prices are simpler to implement, advertise, maintain, and reason about. Dynamic prices are more powerful. They provide feedback that can influence user behavior; they can be used to dynamically change the subset of customers that posses the appropriate budget to pay for accessing a resource; can dynamically adjust the resource exposure and prevent attacks through lack of budget to perform them; and can alleviate the effects of attacks by forcing attackers to spend their budget at an increased rate. Dynamic prices can be a function of several parameters. Figure 6 depicts two such pricing policies, one that depends on time and one that depends on the percentage of the service capacity C of a resource that is currently being used by all customers accessing the particular service. In the capacity-based pricing policy in Figure 6, the resource manager sharply increases the price to access the resource above a certain threshold capacity C_{thres} providing feedback to the customers that the resource is entering in an *undesirable region of operation*. Legal users and attackers will have to pay the increased price to continue accessing the resource.

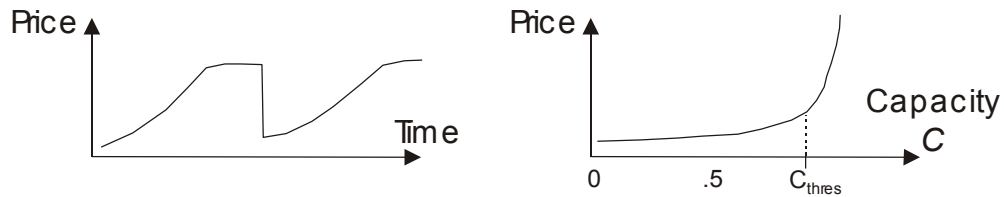


Figure 6: Dynamic pricing policies

The service administrator can provide access to the resource or service it manages for a different price to different users. *Differentiated prices* can play an important role in security. For example, attackers may see a much higher price than normally behaving users. This will deplete their available budget much faster, or in certain cases prevent the attack. Differentiated prices are more complex to implement than uniform prices since they require the resource manager to keep per customer information for the customers it wishes to price differently.

Control of Budget Expenditure

Control of budget expenditure introduces the need to describe and enforce such restrictions. One apparatus that is simple yet particularly powerful in both expressing and enforcing these restrictions is a variant of the leaky bucket mechanism. The mechanism allows per-customer control of budget expenditure restrictions at different time scales. In this variant of the leaky bucket, the service administrator controls the total amount of budget expenditure, the period over which this budget can be expended, and the rate of expenditure of the budget.

The mechanism divides time in equal units of duration T as depicted in Figure 7. Within each time slot of duration T , the customer can spend up to a total of B units of its available budget. The B units are made available to the customer at the beginning of each time slot. If the customer spends the B units of currency before the end of the time slot, he will have to remain idle until the beginning of the next time slot. The customer can keep spending B units within each time slot, up to a total of B_{total} units of budget. The user's expenditure is restricted within a period of time T_{total} .

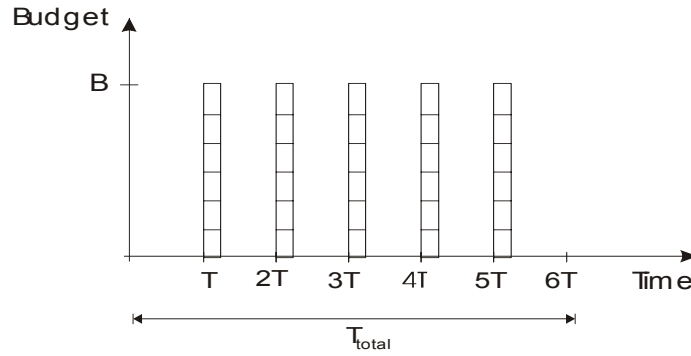


Figure 7: Leaky-bucket budget expenditure control mechanism

There are two important features of this mechanism. First, it guarantees that a particular customer can spend no more than B units of currency at any point in time, independent of the actual budget of the customer. Second, the resource manager can control the rate R of the customer expenditure at any required granularity. This rate is given by $R = B/T$. If the target rate is 10 units of currency per second and the required granularity T is 1 second, then B is set to 10. For a granularity of $T=1/2$ second, B is set to 5.

Detection of Anomalous or Undesirable Behaviors

A final restriction imposed in the power to access resources comes from intrusion detection mechanisms that may block accesses classified as potential attacks or undesirable access patterns. MarketNet can use any intrusion detection mechanism for this purpose. One such detector that may take into consideration the rest of the access-limiting mechanisms outlined above, is developed in the accompanying documents. It is based on analyzing the flow of currencies generated as a result of charging for access to resources and services. The flow of currencies expresses both the expenditure behavior of individual customers, as well as the revenue generation patterns of service administrators. Attacks typically exhibit characteristic expenditure and revenue generation patterns that can be exploited by the detector. For example, a DoS attack manifests itself as a sharp and abnormal increase in the expenditure behavior of the attacker. Similarly, a distributed DoS attack manifests itself as a sharp and abnormal increase in the revenue generation pattern of the resource manager. Upon observing suspicious patterns of access, the detector can limit access to resources, independent of the other mechanisms outlined above.

Complementarity among the Access Limiting Mechanisms

The mechanisms presented above are complementary. Control of budget allocation and control of budget expenditure are two complementary mechanisms that can be used to reason about and quantify security at different granularity and time scales. Control of budget allocation regulates the amount of budget allocated to different entities based on their identity. Allocation can be performed prior to the transaction. Budget allocation can be used to reason about the security properties of whole domains. In contrast, control of budget expenditure regulates, at potentially much finer granularity than budget allocation, the utilization of budget available to an entity. Control of budget expenditure happens at the time when the transaction is taking place. It can be used to reason and provide strict guarantees at the granularity of individual resources.

Prices in conjunction with budget control can regulate who has access to a particular resource and quantify the amount of access to this particular resource. Prices can be used as the parameter with respect to which service administrators control allocation and expenditure of currency. Dynamically adjusting prices can give feedback to clients and influence their access behavior. It is conceivable that if feedback through prices is not important, control of budget expenditure could make the existence of prices redundant.

A.1.3. PREVENTING ATTACKERS FROM BYPASSING THE MARKETNET PROTECTIONS

Preventing an attacker from bypassing the SGM responsible for a resource can be achieved through one of several alternatives. In general attacks of this sort can be dealt with by moving the security interface through which the requests have to go through as close as possible to the resource or service they protect and ensuring that the resource or service accepts requests only through a particular security API. This section briefly reviews previous MarketNet work that is operating-system independent, and outlines the application of transparent MarketNet-based protection to services executing in Linux.

The first protection approach consists of a simple firewall capability depicted in Figure 8. On the left side of Figure 8 the server is protected by a separate firewall host that filters any request not containing appropriate payment. 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 the figure, 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 server code itself to reject any request not coming from the SGM. In both alternatives, the server sends its replies to the SGM that forwards the appropriate reply to the client. Attackers that try to send requests directly to the server are filtered by the site firewall. Attackers that send requests with inappropriate payment are discarded by the SGM.

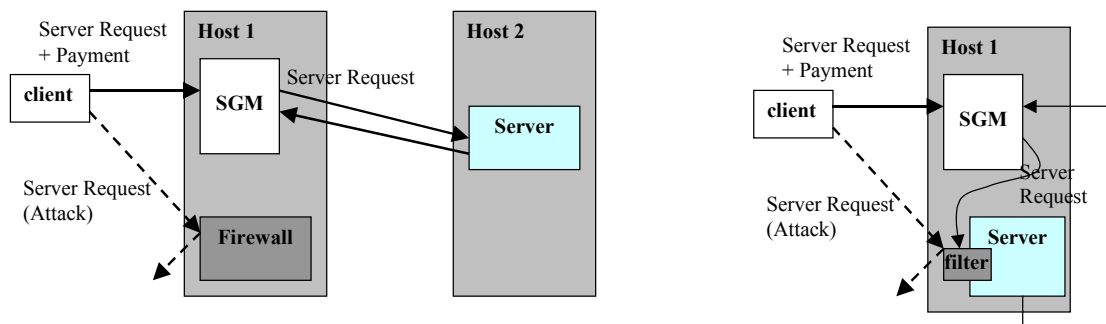


Figure 8: Protecting access to a resource

The second protection approach focuses in cases where the interfaces through which resources are accessed are part of the operating system. Examples include access to resources such as memory, CPU time

and files. Protecting such accesses has been demonstrated in the accompanying documents where the Java Virtual Machine has been instrumented with MarketNet protection mechanisms to protect access to resources such as memory, CPU time and access to files. The JVM can be thought of as a general operating system that controls access to low-level local resources. Instrumentation of operating systems such as Linux or windows with the MarketNet mechanisms should be very similar to the instrumentation of the JVM. The protection results demonstrated in accompanying documents should also apply to the protection of any operating system.

The protection techniques outlined above are operating-system independent. A particularly suitable platform for implementing these techniques is the Netfilter package found in recent Linux kernels. Netfilter allows the specification of simple firewall rules based on TCP and IP fields. Matching packets coming into or going out of a host, can be diverted to a user-defined handling process that can examine them, modify them (e.g., insert or remove payment), and reintroduce them in the kernel on their way from or to the network interface. Figure 9 outlines the sequence of interactions that take place when a client wants to access a general service. Notice that the protection is completely transparent to both client and server. An attacker can bypass it only by taking over the kernel of the server.

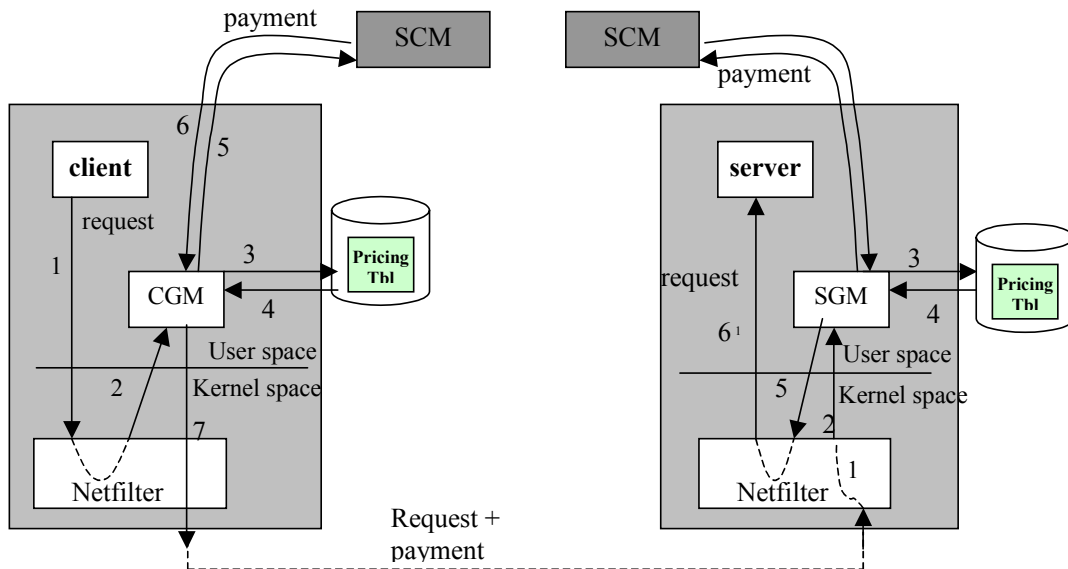


Figure 9: Netfilter-based implementation of MarketNet protection

A.2. PROTECTING REPLICATED DISTRIBUTED SERVICES

Distributed service reliability is often implemented as a set of replicated servers operating in tandem and performing the same computation. Such system can provide correct results even when a minority of the nodes fails. . In particular, a distributed voting protocol can determine the majority result of the servers and publish it as the result of the service. The scheme works especially well in protecting against hardware or software defects because the failure probability of an individual voter (server) can be computed a priori in order to add the required level of redundancy to lower the probability of service failure.

Faults induced by directed attack, however, are not probabilistic in nature and this results in diminishing value of the replication technique. Attackers can attempt to take over the majority of the servers or the voting process itself. Conventional protection techniques provide insufficient protection against compromising individual servers, and almost no protection once these servers are compromised.

MarketNet provides several lines of defense that can make the replication technique an effective tool for critical services, even in the face of malicious attacks. It provides strong protection against compromising individual voter servers or the voting service itself. Furthermore, it allows the dynamic quantification of trust level of each server by associating payments with each vote. Even if an attacker manages to compromise a numerical majority of voting servers, its attack power will be limited by its budget and prices of voting. In addition, the system can automatically respond by adjusting budget and prices and reduce the power of attacks in progress.

A.2.1. TYPICAL REPLICATION SCENARIO

Figure 7 depicts the typical replication scenario where the voting scheme can be applied. There are n servers (S_1 through S_n) that will perform the same computation yielding the results $r(S_1)$ through $r(S_n)$. The computation can be a database lookup, a numerical computation, a sensorial input, etc. The client C is expecting the most accurate result for the computation. One of the key challenges in the scheme is to compute the voting function $v(r(S_1), \dots, r(S_n))$ which will provide the final result r to C .

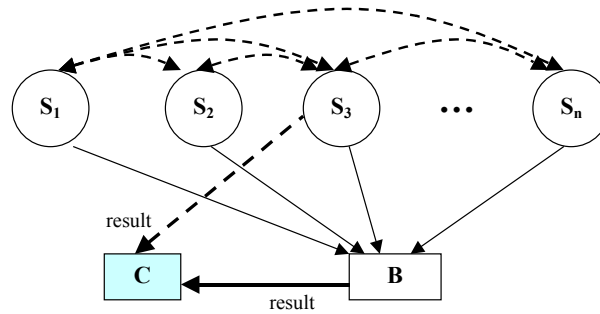


Figure 10: Typical Voting Scenario

The most common scheme will compute v such that r will exactly match the majority of the results $r(S_1), \dots, r(S_n)$. The scheme may be extended in some cases so that the match does not have to be exact, but instead approximated within some accuracy degree ε . The latter is especially useful in sensorial networks where the results from different sensors are unlikely to bring exact matches.

There are several approaches for computing v in a reliable, efficient, and accurate fashion. In the most straightforward approach, voters use a two-phase commit protocol to communicate their results to some intermediate entity B , which will then relate the voting result to C . This scheme has the disadvantage of the vulnerability of B in the commit phase, and its further vulnerability to compromise by attackers. Several schemes have been proposed that can increase the reliability of the straightforward scheme. For example, in the decentralized voting approach, all voters exchange messages, depicted by dashed lines in Figure 9, and each of them independently computes the majority voting result. The client may ask any of the voters (e.g., voter 3 in Figure 9 7) for the result. Alternatively, the client may receive results from all the voters and then compute the outcome of the vote locally. Since this approach produces a correct result when C gets at least a majority of agreeing votes, but not necessarily all votes, this approach eliminates single points of failure in the vote committal phase. The downside of this approach is that the client has to perform the voting computation locally.

A.2.2. PROTECTING AGAINST SERVER COMPROMISES

The first line of defense offered by MarketNet is the protection of the voting servers and of the intermediate entity performing the voting, against compromises by attackers. These are the standard set of protections offered by MarketNet and outlined in Section 0. They include the ability to trace attackers, the ability of individual servers to quantify and tune their exposure to attacks and prevent undesirable use of

their resources, and the automated instrumentation for detecting anomalous behaviors. These topics are covered in more depth in accompanying documents.

A.2.3. QUANTIFYING AND DYNAMICALLY LIMITING THE VOTING POWER OF ATTACKERS

The MarketNet-based voting process allows the system to quantify, limit, and dynamically reduce the voting power of attackers. Voting is based in the following principle: the power of the vote of an individual voter is proportional to the budget the voter is willing to spend on the vote. A decentralized trust management authority assigns budget to voters according to criteria such as trust and knowledge levels of voters. Trust management, expressed through budget allocation, can be totally decoupled from the voting process. This flexibility allows allocation of budget through either local decentralized autonomous authorities and local decisions, or through global coordination. The voting service dynamically evaluates the trustworthiness of individual voters and reflects this through voting price manipulation.

Budget Quantifies and Limits the Voting Attack Power

Budget can reflect any combination of trust criteria to individual voters. For the purposes of this study, budget allocation to an individual node can be assumed to reflect the following criteria: (a) the probability of compromise of the node; (b) the knowledge level of the node; and (c) the probability of random faults of the node.

Budget allocation determines the voting power of a voter. At any point in time, the system can be observed and analyzed. This analysis quantifies the system properties and behaviors. For example, an overseeing authority can determine through this analysis the ability of each individual voter to influence the outcome of a vote or the probability that an attacker can compromise a voting majority. This analysis can be used to maximize security metrics. For example one can tune budget allocation relative to the probability of compromising individual voters, in order to minimize the probability that the voting outcome is compromised; the attacker in this case has to potentially compromise much more than half the voting nodes.

The budget available to a voter limits its voting attack power in two ways. First, the power of the voter to determine the outcome of a vote depends on its wealth. Determining the outcome requires a collection of attacking voters that possess the majority of the wealth. Second, the attacker (or collection of attackers) must spend enough budget to influence the outcome of the vote. The relationship between the budget and the price they have to pay quantifies their ability to maliciously alter the voting outcome. The authority that controls budget allocation can therefore impose limits on the potential damage any attacker or collection of attackers can cause.

System Dynamically Responds to Throttle Ongoing Attacks

Dynamic response to suspicious voters complements the static quantification of and limits of the power of an attacker outlined above. The first dynamic response mechanism increases the prices for potential attackers, based on the level of suspicion. For example, the price rises for voters that voted against the majority in previous rounds. An alternative way of achieving the same results is by decreasing the budget allocated to suspicious voters while keeping prices fixed. This mechanism for limiting the power of potential attackers also assists in limiting the effects of wrong decisions caused by faults or lack of sufficient knowledge.

The second dynamic response mechanism depends on prices determined by supply and demand principles. The more a voter is willing to pay to influence the outcome of a vote, the more this will raise the prices the voter has to pay, limiting the power it has to influence the voting outcome. Notice that under normal (non-attack) circumstances, this mechanism does not in any way penalize voters that vote with the majority.

A.2.4. UNIQUE PROPERTIES OF THE MARKETNET PROTECTION FOR REPLICATED SERVICES

Several unique properties of the MarketNet approach outlined above make it particularly suitable for protecting replicated services.

First, it is orthogonal to the voting architecture and mechanisms used for committing the votes. The protection can be applied equally well to the basic voting scheme, where all votes go through the intermediate entity B in Figure 9, as well as to any other scheme such as variants of the decentralized voting scheme. Furthermore, the MarketNet protection is compatible with any method for committing votes (e.g., two-phase commit).

Second, it allows for dynamic allocation of weights to the voting participants, without requiring any advance or dynamic coordination between entities. This allows for dynamic and rapid adjustment of trust levels of individual voters even in the case of decentralized voting, without any need to dynamically configure individual voting processes in each of the entities that computes the majority vote result. For example, dynamic knowledge that a group of sensors (e.g., S_2 and S_3 in Figure 9) is more trustworthy than the rest in a particular point in time, can be reflected in the allocation of budget to these voters, without having to change anything in the voting process or communicate this decision to any of the other participants.

Finally, it separates trust management and allocation from the voting mechanisms and architecture. Trust management can be delegated to any centralized or decentralized management authority that is totally independent of the voting process. For example, if a management authority decides that one of the voters may have been compromised and therefore should have very low level of trust it can adjust its budget allocation to reflect this knowledge, without requiring any further modifications to the voting process.