MarketNet: Protecting Access to Information Systems Through Financial Market Controls

Y. Yemini, A. Dailianas, D. Florissi

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

G. Huberman

School of Business
Columbia University,
411 Uris Hall
New York, NY 10027
Email: gh16@columbia.edu

Abstract

This paper describes novel market-based technologies that uniquely establish quantifiable and adjustable limits on the power of attackers, enable verifiable accountability for malicious attacks, and admit systematic and uniform monitoring and detection of attacks. These technologies, incorporated in the MarketNet system, establish a financial economy to regulate the trade and use of access rights in information systems. Resources are instrumented to use currency for access control and monitoring, establishing accountability in their use. Domains control access to their resources through resource prices and budgets available to clients. 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 tune their exposure to resource unavailability by purchasing protection through financial-like instruments.

Keywords: Information systems protection, access control through currency, information economy, quantifiable exposure to attacks, financial-like instruments.

[1] Proofs and reorder form should be sent to: Dr. Danilo Florissi, Department of Computer Science, Columbia University, 450 Computer Science Bldg., New York, NY 10027

[2] 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". 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.

1. 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 are increasing with the growth in the range of resources, 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 distrustful accesses to resources and manages trust relationships among users and resources. Attackers can pursue virtually unlimited attempts to compromise a system. They are left completely unaccountable to abuses and crimes that they commit. Access to information systems is therefore susceptible to uncertainties in the availability or resources.

Rapid changes in technologies increase the vulnerability to attackers. First, currently 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. Second, the combinatorial explosion of interactions between new components and existing ones increases exponentially, creating ample possible insecurities. Third, in the absence of a unifying security architecture it is practically impossible for component vendors, or domain administrations to accomplish a coordinated protection.

Domain administrations are thus increasingly exposed to security risks and are unable to control, bound, or even assess this exposure. They require expert manual labor to monitor and correlate access anomalies and detect an attack, typically through off-line non-real-time processes completed hours or days after the attack has been completed. And even when an attack is detected, identifying the source accountable for it can be virtually impossible and requires complex ad-hoc collaborations of multiple expert police forces. And this potential exposure and complexity of protection increases with each change in resources or their configurations.

MarketNet uses market-based mechanisms to provide a novel approach to the protection of large-scale information systems. In MarketNet, resources are instrumented to use currency for access control and monitoring. Clients must pay resource managers, using appropriate currency and prices, to gain access to respective resources. The budget available to an attacker limits its ability to gain access to resources and to cause damage. Domains control the availability of currency to potential sources of attacks and dynamically adjust prices of the resources and services they offer.

Domains quantify their exposure to attacks by strictly controlling the dissipation of currency, along with the prices to access resources. Furthermore, by dynamically adjusting prices they essentially control dynamically varying access disciplines to resources. This access control can take into account variations in the supply and demand, security considerations of the resource managers, and the relative importance of resources to different users.

Currency flows provide uniform resource-independent instrumentation to monitor and correlate access patterns and to detect anomalies. This enables the development of uniform resource-independent intrusion-detection mechanisms entirely based on the statistics of currency flows. Intrusion-detection can be thus automated and accomplished in real-time during an attack. Furthermore, currency carries unique identifiers. A domain maintains full accountability of the entities to which currency has been allocated. A domain can account for sources of each access to its resources. In particular, once an attack has been identified a domain can establish verifiable proof of accountability in tracing its sources.

Client and resource managers quantify and tune their exposure to unavailability of resources by purchasing protection. Similar to the mechanisms used in financial markets to cope with uncertainties in future availability and price, managers purchase risk reduction instruments such as futures and options to hedge against uncertainties caused by loss or congestion. QoS can thus be guaranteed even with uncertainties in the availability of resources. Managers tune their exposure to resource unavailability and optimize their protection based on their needs and budget.

MarketNet mechanisms are structured to admit unlimited scalability and enable protection among mutually distrustful domains organized in a large scale federated system. These protection mechanisms, fur-

thermore, are entirely independent of the underlying resources and can thus be retrofitted into an existing system with minor adaptation of its components.

This paper provides an overview of the MarketNet architecture, mechanisms and operations. Section 2 presents the use of currency for access control in information systems. Section 3 presents the MarketNet financial infrastructure. Section 4 examines the use of currency for dynamically tunable access control. Section 5 provides quantifiable limits on the power of attackers. Section 6 presents monitoring of currency flows to detect and isolate attacks. Section 7develops mechanisms to hedge against uncertainties in resource availability. Section 8 exemplifies the protection MarketNet offers against common attacks.

2. Use of Currency for Access Control

In MarketNet distrustful domains use currency to control access to resources. Domains control the allocation of budget to clients and set the prices for access to resources. Clients pay resource managers, using appropriate currency and prices, to gain access to respective resources. This section discusses the MarketNet system architecture.

2.1 Resources and clients are organized in currency domains

Resources and clients in MarketNet are organized in currency domains. Resources include physical resources such as processor cycles, storage, bandwidth, peripheral devices, sensors as well as higher-level software services such as file storage, name service, database, or web service. A currency domain establishes access protection for a group of resources. It provides administrative infrastructure for imposing domain-level protection policies covering pricing of critical resources, assigning budgets to internal clients, limiting access from external domains, monitoring access to detect intrusion attacks and activating responses to attacks.

Domains use special currency to provide unified and scalable access to their services. The currency is uniquely identified by a *currency ID*. This establishes full accountability in the use of resources by tracing access to resources back to the holder of the currency. To gain access to the resources in a domain, clients

first have to exchange currency of the target domain for their own. Currency of a domain gives the holder the right to access any of the resources in the domain, providing unified, scalable access.

2.2 Domains control allocation of budget and prices of resources

Domains have full control over the allocation of budget to clients. This control is expressed through the currency dissipation policy of a domain. The currency dissipation policy controls who can acquire domain currency, the total currency outflow, the rate of currency outflow, and other parameters. These policies impose strict domain-controlled limits on the access and attack power of any entity wishing to access the domain resources.

Domains also control prices of the resources they offer. Each resource in a domain is priced in terms of the domain currency. Prices, along with their validity period, are advertised in respective price directories. Prices are dynamically updated to reflect various operation parameters such as access control policies and changing demand for a resource. The unique currency identifiers enable the deployment of price discrimination, according to the source of the request.

2.3 A sample transaction

In a typical scenario, depicted in Figure 1, a client belonging to domain *X* wants to access a resource belonging to domain *Y*. The client consults the price directory (step 1) to get information about the price and currency accepted by the service provider. The client then contacts the local bank server (step 2) and requests exchange of his currency for currency accepted by the target service. The local bank server (*X* bank server) contacts the domain *Y* bank server (step 3) and request ex-change of currency. The *Y* bank server authenticates the request and decides whether the request should be granted. If the decision is positive, it records the transaction and passes the *Y* currency to the *X* bank server (step 4). The *X* bank server also records the transaction and passes the currency to the client (step 5). Once the client manager obtains respective currency from its bank server, it can proceed to execute accesses to the service (step 6). Each access will incur a payment collected by the server manager. The revenue collected by the resource manager may now (or at any time later) be deposited to the local bank (step 7). Prices may be updated (step 8) to reflect the new demand level.

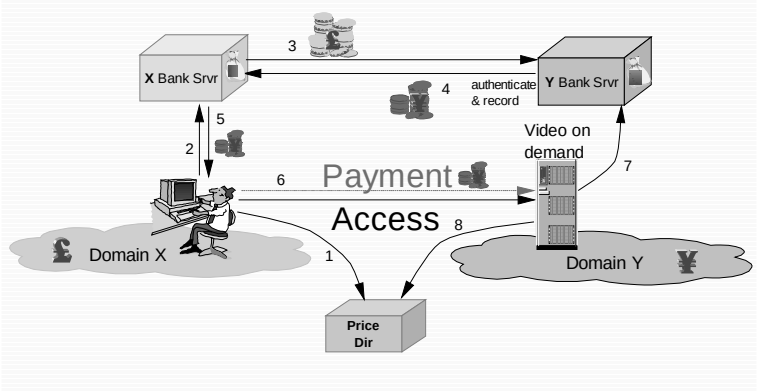

Figure 1: A Sample Transaction

Notice that for efficiency reasons most of the transactions in the above scenario can be performed off-line with respect to the actual transaction. The exchange of currency (steps 2, 3, 4 and 5) are typically performed in advance in anticipation of the clients' needs. Therefore, no significant delay is added to the transaction.

3. The Resource Access Layer (RAL) – A Market-based Protection Kernel

3.1 The architecture and components of the RAL

MarketNet introduces a distributed protection middleware infrastructure, the Resource Access Layer (RAL), for financial access control. The RAL, which is overlaid on existing infrastructure (Figure 2), captures end to end security. The key MarketNet functionality is confined in a minimal kernel that will be rigorously scrutinized against implementation errors or failures.

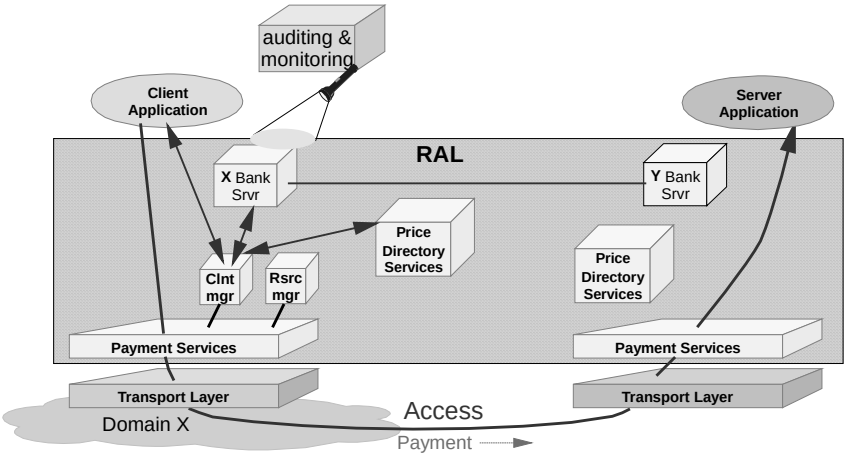

Figure 2: Resource Access Layer (RAL)

The RAL includes several mechanisms. Resource managers are responsible to set the price for a resource, collect payments for its access and deposit revenues with the bank server of the respective domain. Client managers are responsible to manage client budget, obtain pricing information and pass respective payments required to access services used by the client. Bank servers control the flow of currency in MarketNet. They provide accounting, clearing and monitoring of currency flows. Price directories provide pricing information. The payment services layer provides the API through which resources are accessed in MarketNet. It is a thin layer that is logically stacked on top of the transport layer in traditional protocol layering. Payment is performed through a secure money transfer protocol that guarantees that money is not counterfeit, stolen or duplicated (see [2] and [9]).

The most important feature of the RAL is that it kernelizes security, that is, it abstracts protection of resources in the creation of a secure kernel through which resources are accessed.

It is important to note several other salient features of the RAL.

- The RAL mechanisms:

  Are resource and application independent

  Provide incremental extensions of existing components and systems.
- All transactions involving currency flows between managers and their bank servers and between bank servers are secured through encryption and authentication.
- The overheads involved in converting currency among domains and in allocating currency to a client can be minimized through caching of currency. For example, the bank server of domain *X* can cache sufficient currency of domain *Y* in anticipation of requests by clients in its domain.
- Once a client obtains currency, the payment to resource managers involves very minimal overhead.
- The infrastructure is scalable.
- The RAL enables protected domains to operate within an unprotected environment.

The design and implementation of the RAL is of central importance to MarketNet. Due to space limitations, in this section we only present the design of the banking infrastructure, which is among the most important components of the RAL. Examples of resource and client managers are presented in Section 7.

3.2 The Banking Infrastructure

The banking infrastructure in MarketNet is responsible for generating and controlling the dissipation of access rights. Furthermore, it has to clear transactions, audit currency flows, and identify sources of access for attack detection and isolation (as will be explained in Section 6).

Access rights are generated in the form of currency that gives the holder the right to access resources. The banking infrastructure is therefore bound to be of major interest to attackers. Its design has to guarantee that the access-right-power an attacker gains is strictly limited even if the attacker succeeds in taking over a bank. This will provide the basis for the development of quantifiable limits on the power of attackers outlined in Section 5. A minimal set of requirements then in terms of guaranteeing limits in the power an attacker consists of the following. First, the banking infrastructure has to strictly control the amount of access rights an intruder can gain access to when successful in taking over a bank. Second, it has to control exposure to external attacks, i.e., even if an attacker has somehow managed to create or accumulate an unlimited amount of currency of some external domain, this will not affect the security of the internal resources.

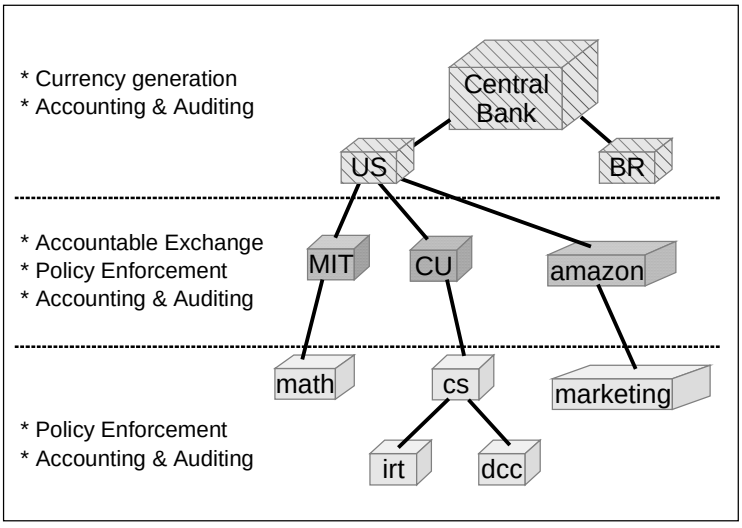

Figure 3: Banking Architecture and Functionality

The banking architecture is hierarchical. It is depicted in Figure 3. Each currency domain has its own domain bank that is responsible for enforcing the domains' currency dissipation policy. For example, in Figure 3 Columbia University has its own bank (*CU*). The Computer Science Department within *CU* has

its own protection policies and currency, enforced by the *cs* bank. For technological reasons (primarily scalability) and legal reasons, different banks in the banking hierarchy have different functionality:

The *Central Bank*, along with a few other banks, is the only ones responsible for generating currency for children domains. They also perform accounting and auditing of transactions.

A number of banks such as *MIT, CU* and *amazon*, typically high in the hierarchy, are the only legally accountable entities authorized to perform exchanges of currency. They also enforce the currency dissipation policy of the currency domain and perform accounting and auditing of transaction. These banks are entitled to perform exchanges on behalf of their descendent banks.

The rest of the banks, such as *cs* (which could be thought of as *CU.cs*), *dcc* (or *CU.cs.dcc*), *marketing*, etc., are responsible for enforcing the currency dissipation policy of the currency domain they represent. They also perform accounting and auditing of transactions. They forward requests for currency exchange to their parent banks.

Currency for a domain, e.g., *CU*, is generated only upon the deposit to the *Central Bank* of an equal amount of currency from another currency domain, e.g., *amazon*. The newly generated currency contains the name of the currency domain it is intended for, along with a unique currency identifier, and is signed by the *Central Bank*. The domain bank (*CU* in this case) further signs it prior to exchanging it for other currencies (*amazon* in this case). Even if the *Central Bank* is conquered, no one will be able to generate valid *CU* currency.

All of the above transactions are strictly authenticated. Notice that for reasons of efficiency, authentication keys can be cached locally and updated periodically. Caching is possible because of the limited number of banks that can perform exchanges and the limited number of keys needed at each level of the hierarchy.

All the banks are implemented using stripped down versions of operating systems that only support the functionality and ports necessary for the banking infrastructure. This minimizes the exposure to attackers. Furthermore, they monitor currency flows to detect breaches in other banks.

A sample transaction

Figure 4 depicts a sample transaction, where client *A* in the *cs* domain acquires *amazon* currency to access service *B*. Client *A* contacts her local bank (*cs*) and asks to exchange part of her budget to *amazon* currency (step 1). *cs* cannot perform exchange, so it forward the request to *CU* bank (step 2). *CU* contacts *amazon* bank (step 3). *Amazon* checks if granting the request would violate its currency dissipation policy. If not, it records that it is passing local (*amazon*) currency with unique identifier *1234* to *CU*, signs the currency and passes it to *CU* (step 4). *CU* passes the currency to *cs* (step 5), and records the fact that it did so. *cs* passes the currency to *A* (step 6) , and records the fact that it did so. *A* now accesses the service *B* and pays for the transaction (step 7). *B* may now, or at any time later, deposit the revenue with its local bank (*amazon*), that will credit its account. Off-line from the transaction, *amazon* contacts the *central bank*, deposits the *CU* currency it received and asks for generation of *amazon* currency. *Central bank* generates new *amazon* currency, with unique id *9876* and passes it back to *amazon*. Notice that most of the transactions in steps 1 through 6 are performed in advance in anticipation of future demands, therefore insignificant delays are added to transactions.

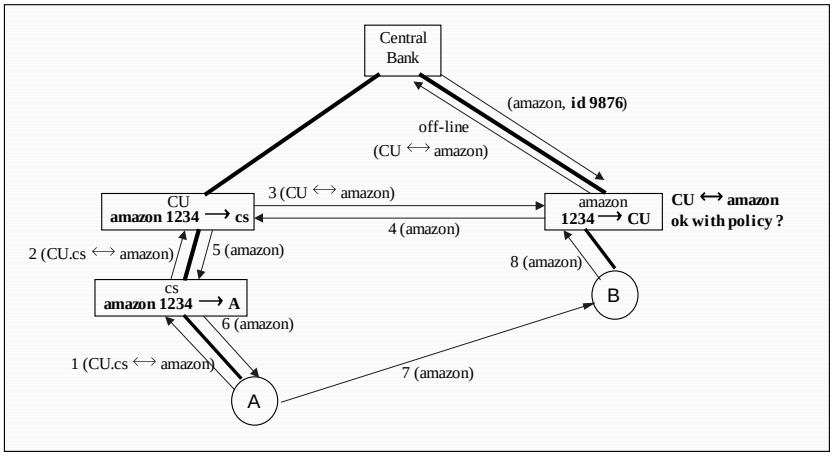

Figure 4: A Sample Transaction

4. Prices/Budgets Establish Dynamically Tunable Access Control

Pricing of resources along with limits on available budgets of clients are used to dynamically control access to a specific set of clients.

Raising the prices ensures that only qualified clients (those that have sufficient budget) can access them. Furthermore, currency identifiers enable additional price discrimination techniques. Budget and price discrimination can achieve a continuous spectrum of limits imposed on the use of a resource, based on the source domain of a request.

Prices are adjusted to reflect changes in the supply and demand for a resource. A loss of a resource due to failure or attack reduces the supply, resulting in increased prices. An increase in the demand for a resource has similar effect. Rising prices due to loss or congestion automatically cause clients to redirect their demands to backup less expensive replicas. Reduced availability results in rising prices of the replicated resources. Rising prices create a natural selection process where applications automatically adapt to resource availability and obtain access to alternate resources according to their intrinsic priority captured by their budget. High-priority clients can apply their budget to continue and obtain high QoS, while low priority clients are priced-out. Thus a loss results in graceful selective degradation of services that optimizes the balance between available resources and demands.

Prices are set and adjusted taking into account the currency dissipation policy imposed by the domain. It is conjectured that most access control policies can be formulated as a combination of currency dissipation policies and price control policies.

5. Domains Set Quantifiable Limits on Power of Attackers

Access through currency provides a framework to quantify and measure exposures to attacks. Domains use their control over the budget available to clients to and the prices of resources to:

- Control and fine-tune their exposure to attacks.
- Dynamically adjust exposures in response to emerging risks.

Exposure to attacks is expressed as quantifiable bounds on the damage that can be caused by attackers. These bounds are described parametrically and enable tuning of resource security. They take into account the budget already allocated to clients, the budget allocation policy parameters such as the rate of alloca-

tion of budget, the maximum total budget that can be available to clients at any time, and the characteristics of the pricing mechanism.

Due to space limitations we are only going to provide an example of the kinds of bounds that can be developed in MarketNet. For more information, please refer to [3]. The example refers to the duration of a denial of service attack and shows how the pricing mechanism provides a means to convert a *fixed* budget (belonging to a specific client or a coalition of clients potentially residing in different domains), to a much lower *effective* budget.

In general it is hard to parameterize what constitutes an attack. For the purposes of demonstration of the bounds in the denial of service attacks, we introduce the notion of *desirable* regions of operation of a resource. The *desirable* region of operation is resource-dependent, and in general refers to the region of operation specified by the resource manager, where specific QoS constraints or other considerations are satisfied (e.g., the average incoming rate to a switch should be controlled to provide low delays and loss). Assume the purpose of attacking a resource is to move it to an *undesirable* region of operation. Control of prices can force the operation of resources within a *desirable* region of operation. The price of the resource should reflect its reluctance to operate in the *undesirable* region. Should the attacker or coalition of attackers desire to sustain the attack, they would see a continuously increasing price to access the resource, forcing them to spend their budget at an increasing rate to sustain the attack. The formula describing the duration of the attack T_{attack}, with a fixed budget B_{attack} belonging to an attacker or coalition of attackers, assuming that all users see the same price to access the resource is the following (for derivation see [3]):

$$\int_0^{Tattack} C_{others}(t)\, dt = T_{attack} * C_{thres} - B_{attack} / P(C_{thres})$$

In the above formula, C_{thres} is the maximum capacity the system can offer without degrading the QoS seen by clients, $P(C_{thres})$ is the price seen by users at this capacity, and $C_{others}(t)$ is the capacity requested by users other than the attacker. Given a target maximum duration of attack T_{attack}, and the budget allocation

policy of the domain, expressions such as the above can be used by the service providers to dynamically adjust pricing policy and thus adjust their exposure.

Similar formulas can be used to design and dynamically adjust pricing policies when price discrimination is deployed, or to quantify other limits on the power of attackers.

6. Attacks are Detected and Isolated by Monitoring Currency Usage

Currency flows provide uniform resource-independent instrumentation to monitor and correlate access patterns and to detect anomalies. It enables the development of uniform resource-independent intrusion-detection mechanisms entirely based on the statistics of currency flows. Intrusion-detection can be thus automated and accomplished in real-time with an attack. Furthermore, currency carries unique identifiers. A domain maintains full accountability of the entities to which currency has been allocated. A domain can account for sources of each access to its resources. In particular, once an attack has been identified a domain can establish verifiable proof of accountability in tracing its sources.

Currency flows provide a good way to model temporal spending behaviors of clients and patterns of resource access to classify activities into those that are legitimate and those that seem suspicious and hence warrant further inspection and authorization. Monitoring the budget spending patterns of clients can thus provide a resource independent technique to detect attacks. Consider as an example Figure 5. This figure is based on real access traces and depicts the amounts spent by different users on accessing network services. For this particular experiment, only access to ports was priced. The left side of the figure depicts the spending patterns of a collection of users with similar total spending over a period of time of 6000 sec. There are three users with similar spending patterns very close to time 0. Their behavior is suspicious since they spend a big amount of budget in a very short period of time. A closer look at their behavior on the right hand side of Figure 5, reveals that in reality only one of them exhibits an unusual spending pattern, which indicates the actual *syn* flood attack the specific user was performing at the time.

Similar techniques are applied in observing the revenues generated by specific services. Anomalies in the revenues of a single service or in the correlation of revenues generated by services in a domain, are good indications of attacks.

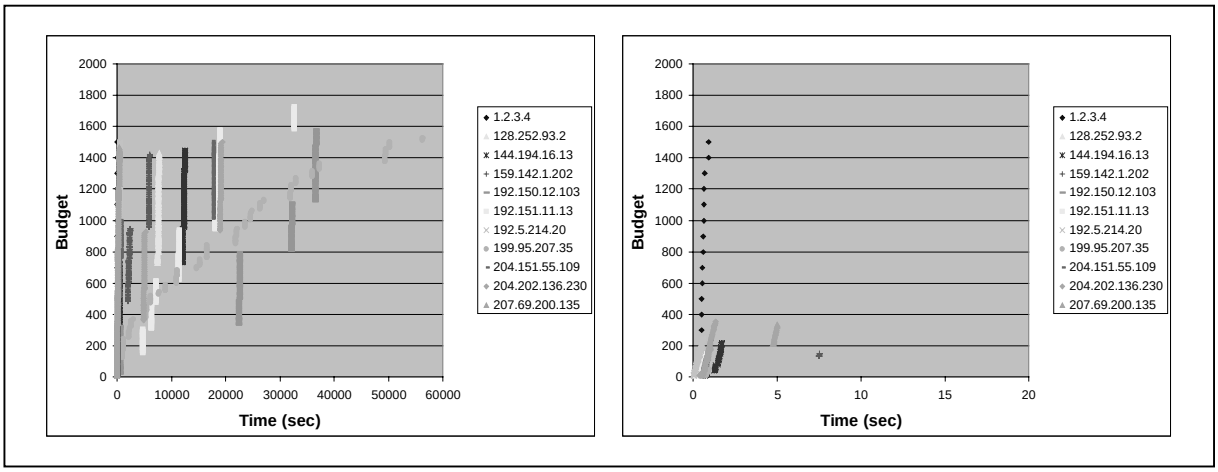

Figure 5: Observing Spending Patterns

7. Financial Instruments Provide QoS Guarantees

Financial markets have created risk reduction instruments such as futures and options, used to purchase resources in the future and hedge against intrinsic uncertainties in the price and the availability of resources. Analogous instruments can provide protection against loss of resources due to congestion or attacks in information systems. Client managers acting on behalf of applications purchase protection in the form of instruments, subject to their budget, to guarantee their QoS requirements are met even when uncertainty in the availability of a resource is involved. Similarly, resource managers rationally purchase backup resources or replicate their own resources to optimally guarantee continuous provision of services to their customers. The QoS guarantees provided through the use of instruments are optimally tailored to the needs and budgets of the acquirers.

We demonstrate the concepts, through the use of instruments to provide QoS guarantees that depend on bandwidth availability. Bandwidth is traded through the use of financial instruments similar to those in

financial markets. Two types of instruments are used: forwards contracts and options on these contracts. The mechanism is not restricted to these instruments. Consider as an example a user in Figure 6 who wants to participate in a teleconferencing with a remote site at noon. There are two kinds of risks the user is facing. First, congestion for link bandwidth at noon jeopardizes transmission quality. To avert this risk, the user buys forwards contracts on link A that guarantee his access to the resource (bandwidth), independent of the congestion level. Second, link A may go down. To avoid this risk, the user buys options contracts on bandwidth on link B. If link A goes down, the options are exercised to purchase bandwidth on link B.

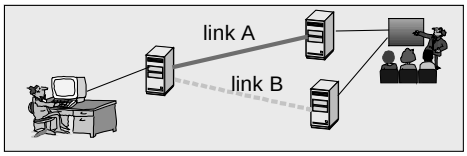

Figure 6: Financial Instruments Provide QoS Guarantees

In the pilot implementation the participants in the market are one service provider and several customers. Both can assume the role of buyers and sellers. Offers and requests along with the corresponding prices are posted in a bulletin board, which plays the role of a broker that matches the requests and the offers. The bulletin board implements the *primary market* that trades forwards contracts offered by the service provider and *secondary market* that trades options on forwards contracts offered by resellers.

There are two kinds of customers: customers with strict QoS guarantees and high budgets (rich customers) and customers with loose QoS requirements and low budgets (poor customers). Customers try to maximize their utility function subject to their budget constraints, while the service provider tries to maximize its revenue and guarantee that the maximum number of users is satisfied. We have experimented with simple strategies that are consistent with the selfish optimization objectives of the participants. The simplest such strategy for the service provider is one where she is trying to maximize the utilization of the bandwidth. The provider objectives are expressed through the pricing policy that assigns prices to reflect demand statistics for corresponding bandwidth utilization in the past. Prices are dynamically adjusted to reflect the level of the current (real) demand. Customers interested in buying guaranteed

service participate in either market as buyers, while customers who have previously purchased surplus guaranteed service participate in the secondary market as re-sellers. Poor customers try to send their traffic using best-effort service, which is cheaper than guaranteed service. They buy options to use guaranteed bandwidth when they predict there will not be enough bandwidth to guarantee their QoS constraints. They resell their options through the secondary market as soon as they realize they are not going to exercise them. Rich customers try to acquire the required bandwidth to strictly meet their QoS requirements. They are always willing to buy at the market price set by the service provider. They buy enough bandwidth to absorb any uncertainty in their traffic characteristics; as soon as they realize they are not going to use some of the bandwidth they have previously bought, they will try to resell it in the secondary market as options. The price of the options is calculated through the Black-Scholes formula (see for example [4]). Our studies of purchasing access protection through instruments similar to those in financial markets have demonstrated several advantages. First, provision of differentiated, user-defined guarantees. These guarantees are optimally tailored to the user needs and budget. Second, better utilization of resources compared to systems that do not provide the ability to resell resources. Third, decreased cost for the users. Fourth, dynamic deployment of new resource management algorithms that are reflected in the pricing strategy of the service provider.

8. Examples of MarketNet's Reaction to Attacks

In this section we take as an attack example the famous "worm attack" ([10]) and show how MarketNet would have reacted to it. The purpose of the worm was to spread itself, i.e., to break into systems, install and locally run itself. To achieve this goal it exploited flaws in utility programs based on BSD-derived versions of Unix. The worm program eventually spread itself to thousands of machines around the Internet, and disrupted normal activities and Internet connectivity for many days. The disruption was caused by the worm's heavy use of system resources in order to break to other systems. The worm was not destructive in nature, i.e., it did not delete user files or try to cause other such disruptions, but it has to be noticed that if its intention were to do so, nothing would have prevented it.

The worm spread using three techniques:

1. Exploiting a bug in the *gets* standard C I/O library used by the finger daemon (*fingerd*). The *gets* call stores its input into a buffer without doing any bounds checking. The worm overran the buffer boundaries, gained access to privileged memory regions, and installed and executed itself locally.
2. Using the *debug* option in *sendmail* servicing the SMTP port. The debug feature enables testers to run programs on remote machines to display the state of the remote mail system without sending mail or establishing a login connection.
3. Guessing passwords and exploiting trust between hosts. Once present in a system the worm would try to guess user passwords, break into their accounts, and exploit trust between hosts (e.g., hosts found in /etc/hosts.equiv and /.rhosts) to remotely execute into those systems as well. Password guessing is a computationally very intensive operation.

Using any of the above methods, the worm would successfully spread without leaving any trace of where it came from.

MarketNet can protect against several features of worm-like attacks. First, the prospective attacker would have to pay to use *sendmail* or *fingerd*, leaving an unforgeable trace of the originator of the attack. Second, using system resources is not free. To perform password guessing the process would involve heavy system resource utilization. Monitoring of the budget usage at the conquered account domain would soon trigger alarms due to the unusual behavior. Furthermore, the amount of damage (e.g., overloading system resources) the process can achieve is limited by the budget available to it. Mechanisms to impose restrictions on the budget available to processes are currently under investigation in MarketNet.

MarketNet protects systems without eliminating software bugs. It assumes that software bugs are always very likely to exist and creates a layer of protection that is independent of the correctness of software.

The worm attack is one of the most difficult attacks to handle and shows some of the limitations of MarketNet. These limitations are not particular to MarketNet. The limitations under consideration stem form the fact that software implementation bugs may allow intruders to impersonate legal users of systems and

therefore gain the same privileges the legal user would have. We are currently investigating how MarketNet can efficiently react in the following scenarios:

Assume that the worm had destructive intentions. Budget enforcement along with usage monitoring in MarketNet would limit the scope and the extent of the damage. We are currently investigating price discrimination techniques that may be able to limit such attacks by making resources very expensive when the process does not normally use it. For example, deletion of files would be very expensive for unknown processes, which will not have enough money for the attack.

In a worm-like attack, the attacker manages to impersonate the owner of an account. Even when this happens, it should not be equivalent to getting hold of the budget of the conquered account. One of the mechanisms to break this equivalence is usage monitoring. Abnormal access patterns can be restricted providing adjustable limits on the amount of damage a malicious or faulty processes can cause. A second mechanism under investigation is the separation of budgets available on a per process and/or per task basis. The tradeoff in this case is protection level vs. ease of use of the system.

9. Conclusions

Market-based technologies can provide an effective solution to the protection of information systems. MarketNet develops unified, systematic, market-based technologies to provide scalable and tunable access control for large multi-domain networked systems. These technologies enable unified monitoring and correlated analysis of resource access to detect intrusion attacks, isolate the sources of attacks and respond quickly to control its damages. MarketNet develops mechanisms to protect critical network services, based on their quantifiable value to users, and assure their continuous availability through failures or attacks based on user's priority.

In summary, some of the key ideas in MarketNet are the following:

Currency is used to provide unified, scaleable, resource-independent access control to resources and services and account for their use.

Resources and clients are organized in currency domains. Each domain has its own currency. Resources are instrumented to use currency for access control and monitoring. Clients wishing to access a resource must pay in currency acceptable to the domain that owns the resource.

Organization in currency domains can limit the spread of faults and attacks.

Domains control exposure to attacks, by controlling allocation of budget to clients and prices of resources. This exposure is quantifiable and can be dynamically adjusted in response to emerging risks.

The banking infrastructure strictly controls the dissipation of access rights, and guarantees strict limits on their acquisition.

Currency carries unique unforgeable identifiers that can be monitored and traced back to the holder. Currency identifiers establish verifiable accountability on the use of resources.

Currency provides a resource-independent instrumentation to monitor and correlate access patterns and to detect intrusion attacks through automated, uniform analysis of anomalous currency flows.

Financial instruments are used to hedge against uncertainties in the availability of resources. Managers use them to tune their exposure to resource unavailability and optimally purchase protection according to their QoS needs and budget.

These mechanisms are resource-independent, and admit scalability for very large systems consisting of federated domains operated by mutually distrustful administrations

Bibliography

[1] S. Clearwater, editor, Market-based Control of Distributed Systems, World Scientific Press, 1996.

[2] A. Dailianas, and Y. Yemini, A Protocol for Secure Financial Transactions, Technical Report, Distributed Computing and Communications Lab, Dept. of Computer Science, Columbia University.

[3] A. Dailianas, Use of Currency for Access Control in Large-scale Information Systems, Ph.D. Thesis Proposal, Department of Computer Science, Columbia University, Sept. 1998.

[4] J. C. Hull, Options, Futures, and Other Derivatives, third edition, Prentice Hall.

[5] J. Kurose, M. Schwartz, and Y. Yemini, A Microeconomic Approach to Optimization of Channel Access Policies in Multiaccess Networks, in: Proc. Of the 5th International Conference on Distributed Computer Systems, Denver, Colorado, 1995.

[6] J. MacKie-Mason and H. Varian, Economic FAQs About the Internet, Journal of Economic Perspectives, vol. 8, no. 3, pp. 75-96, 1994. Reprinted (with revisions) in Internet Economics, J. Bailey and L. McKnight, eds. Cambridge, MIT Press, 1996.

[7] J. MacKie-Mason and H. Varian, Pricing the Internet, in: B. Kahin and J. Keller, editors, Public Access to the Internet, ACM, Boston, Massachusetts, May 1993.

[8] J. Sairamesh, D. Ferguson, and Y. Yemini, An Approach to Pricing, Optimal Allocation and Quality of Service Provisioning in High-speed Packet Networks, in Proc. of the Conference on Computer Communications, Boston, Massachusetts, April 1995.

[9] B. Schneier, Applied Cryptography, second edition, John Wiley & Sons, pp. 139-147.

[10] E. Spafford, The Internet Worm Incident, Technical Report CSD-TR-933, Department of Computer Sciences, Purdue University, Sept. 19, 1991.

[11] W. Walsh, M. Wellman, P. Wurman, and J. MacKie-Mason, Some Economics of Market-Based Distributed Scheduling, in: Proc. of the 8th International Conference on Distributed Computing Systems (ICDCS-98), Amsterdam, the Netherlands, May 1998.

[12] Y. Yemini, Selfish Optimization in Computer Networks, Proc. of the 20th IEEE Conference on Decision and Control, pp. 281-285, San Diego, CA., Dec. 1981.