Market-Based Protection of Information Systems

Apostolos Dailianas
Department of Computer Science, Columbia University,
450 Computer Science Bldg., New York, NY 10027
Email: apostolo@cs.columbia.edu

The Challenge

We propose the development of novel market-based mechanisms to ensure the systematic, quantifiable and predictable protection of large-scale information systems against attacks. Success in this research will result in scalable and resource-independent mechanisms that

1. establish quantifiable and tunable limits on the power of attackers to access or damage critical information systems resources
2. establish full accountability among separately administered and mutually distrustful domains; and enable rapid tracing and isolation of attack sources
3. provide resource-independent instrumentation to monitor resource access and to detect intrusion attacks automatically
4. provide quantifiable hedging against loss of critical resources due to attacks or failures and graceful recovery that reallocates resource access based on quantifiable priorities

Technical Approach

Resources and their clients are organized in currency domains. Resources include both physical resources such as CPU cycles, storage, bandwidth, I/O devices or sensors as well as higher-level services such as operating systems calls, file systems, name servers, database or web servers.

Domains use currency to provide unified, scalable, and traceable access to their services. Each domain has its own currency. A client accessing a resource must pay for its use with the currency of the domain that owns the resource. To acquire foreign currency, clients first have to use part of their budget and exchange it for the desired currency. Access is thus limited by the budget available to the client and by the price of the respective resource. Each domain fully controls access to its resources through several parameters: the price of a resource; the budget allocated to a given client (whether an internal client or an external distrusted domain); and the rate at which currency is provided to a given client. Currency carries unique unforgeable identifiers that can be monitored and traced to establish full accountability of the source of an access.

The combination of prices and available budgets provides uniform resource management, dynamically tunable access control, and resource protection. Prices are dynamic. They rise with increase in demand and fall with increase in supply. With a fixed supply, a rise in price indicates an increase in demand, and suggests to current or potential owners of resources to spend part of their revenue to increase supply. If supply does not increase, or decreases due to a failure or attack, higher prices moderate demand, and allocate the scarce resource to those valuing the resource most. Protected resources have extremely high access price; only qualified clients that have been given the appropriate budget can access them. In case of shortage in supply due to loss, attacks, or congestion, access is provided to high-priority customers, i.e., those willing to pay the high prices and possessing the available budget. The rest are priced out and automatically redistribute their demand to similar services. Protected resources can limit access to a specific set of qualified clients that have been given the appropriate budget to access them. In case of attacks rising prices force attackers to spend their budget at an increasing rate to sustain the attack. While rising prices severely limit the power of attackers by essentially lowering their budget, high priority (high budget) customers are still able to access the resources under attack.

Resource managers use value of resources to rationalize replication. Provision of resources and services generates revenue that is directly related to the demand for and the importance of the

provided good. Resource managers quantify the loss of revenue due to a loss of a resource. They use this estimate to hedge against loss by investing current revenues to insure future availability. The services most valuable by their clients are provided with the highest redundancy.

The flow of currency among clients, resource managers and different domains is managed by respective bank servers. Bank servers maintain accounts, enforce budget constraints, clear transactions among clients and resource managers, and monitor currency flows. The bank servers are hierarchically organized to admit scalable organization of nested domains with independent resource protection and control policies. Bank servers maintain accountability among independent domains by controlling currency flows among them. The banking system is protected through strict traditional mechanisms as well as market-based mechanisms.

Resource access is monitored to automatically detect intrusion attacks. Attack sources are identified and isolated. Monitoring is done in much the same fashion as in transaction processing systems. The unique currency identifiers are used to correlate resource accesses to sources of access. Currency flows provide a good way to model temporal 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. Once an attack has been identified, identification through the currency identifiers isolates the source of the attack. This knowledge is rapidly propagated to other domains to avoid further spread of faults or attacks.

The power of attackers is quantifiable and tunable. A first limit to the attack power of a network entity (where an entity could be a set of clients or a set of domains) is imposed by its available budget. The buying power of any entity is limited by the income it generates through the services it provides to the rest of the world. Spending within the available budget is enforced by the hierarchical banking infrastructure. A second limit is imposed by the currency policies enforced by domains. The budget a domain provides to some entity, along with the rate at which it provides it, imposes a strict limit on the access (and attack) power of any entity. A third limit is imposed by the pricing mechanism. Prices provide 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. For example, if the purpose of attacking a resource is to move it to an "undesirable" region of operation, then the price of the resource should reflect its reluctance to operate in that 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 exhaust their budget at an increasing rate. Knowledge of the specific pricing policy in this case can provide analytical upper bounds on the duration of attacks achievable by given collective budgets.

An example: The Worm Attack

As an example consider the “worm attack”, one of the hardest to prevent and react to. The purpose of the *worm* was to break into systems, install and locally run itself and spread to other systems. Three techniques were used to do so: a. exploiting a bug in the *gets* C I/O library used by the *fingerd* daemon; b. exploiting the *debug* option in *sendmail* servicing the SMTP port; c. guessing passwords and exploiting trust between hosts. If MarketNet were in place, the attacker would leave an unforgeable trace by paying to use *sendmail* or *fingerd*. Furthermore, the password guessing attack would soon be detected and isolated since it involves heavy system resources utilization. Monitoring of the budget usage would 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. Notice that we make the worst-case assumption that the intruder manages to get hold of the budget of the conquered account. 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.