Administration in a Multiple Policy/Domain Environment: The Administration and Melding of Disparate Policies

William R Ford
William Ford, Consulting
13 Coach Road
Billerica, MA. 01862-2508
wrf@world.std.com

Abstract

New standards for trusted systems propose multiple security policies and multiple policy domains. My experience building a Mutipolicy Machine prototype illustrated that multiple policy domains and complex policies push current policy administration techniques, tools, and user interfaces beyond their limits. This paper proposes a holistic aproach to policy administration consisting of human-intuitive user interfaces for defining policies, a PolicyBase (a term I am introducing to describe a KnowledgeBase focused on the rules and data required to describe the policies to be administered and acted on) for storing them, and intelligent tool programs that allow the adminstrator to anticipate the impact of policy changes and interactions.

1. Introduction

Information overload makes even relatively simple policy administration ineffective and increasingly difficult to automate even in environments governed by the most basic of policies. The additional complexity of administration in a multiple policy (and/or a multiple domain) environment pushes available technology and techniques beyond their limits. The increasing sophistication required of computer users, administrators, and systems, requires new paradigms for computer policy data administration management tools and administrator interfaces. Mathematical statements of policies are usually helpful to formalists and algorithm developers .This paper proposes an intuitive approach to presenting and maintaining policy rules and data which is a better match to the "intuitive" nature of human cognition, while still including the mathematical analysis needed to interpret policy rules in an automated fashion. Present maintenance software and hardware tools force people to do an increasingly complex set of tasks with counter-intuitive data presentation and management interfaces. Any attempt to automate the administration and enforcement of policies formed by and for human beings stumbles on the hard to define nature of human cognition. Predicting the effects and interactions of new or modified policies is extremely difficult.. Simulated enforcement of new and modified security policies should improve the likelihood that new policies will accomplish the desired goal. A suite of sophisticated, intelligent, and interactive tools is required to automate policy administration to support effective automated decision making based on policies specified in a PolicyBase (a KnowledgeBase focused on the rules and data required to describe the policies to be administered and acted on). The Holistic Administration Tool (HAT), comprised of human intuitive user interfaces and many intelligent tool programs, proposed by this paper would provide the necessary functionality to allow the effective administration and use of disparate policies.

The function of modern computer systems is a complex task requiring a knowledgeable administrator using a large number of software tools. Present concerns about

0-8186-7318-4/95 $4.00 © 1995 IEEE

privacy[1], protection of data for national security and public safety reasons[2], and/or the guaranteeing that a computer system supply a particular service or data all come into conflict as different polices are administrated and enforced. The reader should keep in mind that the policy maker, the policy automator, and the policy administrator are different roles performed by very different people with different skill sets who need different tools. The policy maker is a decision maker who makes or changes policies. The policy automator defines the policies in terms that the computor can understand and makes sure that they work and interact properly. The policy administrator performs the operational functions of installing policies and auditing their enforcement. However, in our prototype, all of these roles were performed by the policy administrator.

This project explores a proposed paradigm shift for computer security technology. The shift from operating systems with a monolithic, built-in security policy which is the foundation of all assurance to a multiple, independent, contradictory grouping of security policies that need to be evaluated independently of the machine architecture is the underlying concern. Adding system management of even relatively simple policies, such as MAC or DAC security policies, makes the task even more complex. An incremental approach was instituted in our prototype; forming the framework first then adding policies of increasing complexity. Attempting to enforce more sophisticated policies within the prototype pushed the management task to the edge of what is presently technically possible. Unpredictable technological change [3] can make present policy administration and enforcement tools less effective, or even negate them completely. Interpreting and enforcing policies with unrelated goals from different domains, with different data requirements goes beyond the edge of technical possibility into the realm of "mission: perhaps impossible." [1] This paper discusses a conceptual approach that can take system policy administration and enforcement in a multipolicy environment from the technically impossible to the realm of technically possible.

1.A. The Problem - An Intuitive Impression

The administration of a computer system is a complicated endeavor fraught with pitfalls caused by

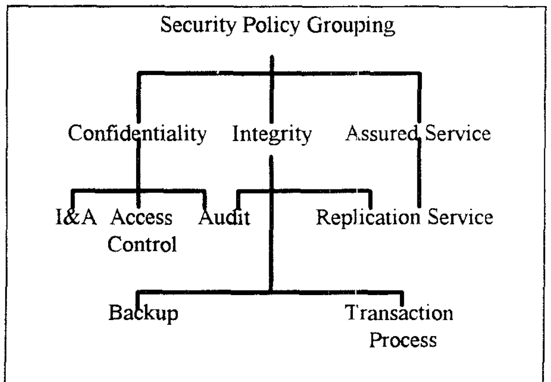

Figure 1 Policy Interrelationships

inadequate software and/or inadequate operator or administrator training. When enforcement of policy becomes part of operating system responsibility the administration burden increases enormously, made more difficult, if not impossible, by the presence of unrealistic (i.e., unmanageable) policies, even those within the same domain. [2] Policies within a domain tend to have one or more goals in common, but may have conflicting goals. The melding of policies to enforce a decision or particular goal between and across multiple domains can be extremely complicated, with conflicts that cannot be clearly resolved. The administration and melding of disparate policies (even with the support of decision making software to deal with and control information overload) is non-trivial, especially when such administration impinges on areas of the human psyche where decisions are made in a gestalt fashion (where the person making the decision is not consciously aware of all the factors that lead to the final result).

For some goals even disparate policies can be melded to arrive at a decision. For example, if the goal is to provide a service at all costs regardless of other polices then any disparities (conflicts between policies) will be ignored to provide that service. If, on the other-hand, the goal is to prevent an entity (as defined by the system security administrator) getting access to a certain resource (information or functionality) based on what that entity knows and its roles then arriving at a decision to grant or deny access to that resource is non-trivial. Present computer systems and software restrict access in relatively inflexible and not very sophisticated ways, with an administrator making a relatively coarse decision manually (i.e. assigning a user a Top Secret Clearance so they can get all the information they want because "they" are on the mission commander's staff) even if "they" don't really need access to

[1]. The most common goals being security, safety, or assurance.

[2] By implication, policies are usually formulated within a framework, or given extent, often referred to as a domain.

the information. This relatively coarse and inflexible method of making decisions is error prone and can not handle the complex issues that must be dealt now, and in the future.

Figure 1 shows a relatively simple policy grouping relational hierarchy. Even this simple policy has internal conflicts and non-trivial interrelationships. Confidentiality, integrity, and assured service are goals that conflict with each other. How their conflicts are resolved would most likely be derived from the results desired by the policy maker (i.e., if confidentially were all-important, then integrity and assured service would lose out). If assured service were the primary goal of the policy (say for air traffic safety) then confidentially would fall by the way-side. Operationally, the elements required to support the policy in themselves require "operational" policies about how they are to interact, what decisions should be made, and how they should be made. For example, policies governing audit processing might conflict with backup processing or with the goal of the "main" policy (confidentially, for example). Many other possible conflicts can be found in this figure, given a particular desired goal and the operational policies to support that goal.

1.B. The Problem - As Experienced

The interaction of unrelated policies from separate domains creates administrative havoc that is difficult to resolve. The actual goal of the policy is irrelevant. Cross couplings and interactions between the description of policies and their desired goals versus what the wording implies, makes interpretation by humans and software programs difficult and the maintenance of integrity between the data elements a technical challenge of the highest order. The implementation and use of the administrative interface for the MultiPolicy Machine (MPM) prototype[4] ran head-long into this nightmarish situation.

The MPM Administration Tool (MPM-AT) was an OpenWindows based GUI front-end to a Object Oriented library of database service modules[5]. Because the main goal of the project was to demonstrate an actual MPM Decision and Enforcement engine (MPM-DE) the administration tool was considered a very minor, but necessary, tool that had to be implemented to insure a PolicyBase for the MPM. Figure 2, below, is a preliminary interface design drawing that implies the relative importance given to the administration functionality, versus the other modules, during the early stages of the project. It came as a shock to realize that the administration tool was at least as important as the decision and enforcement portions of the MPM.

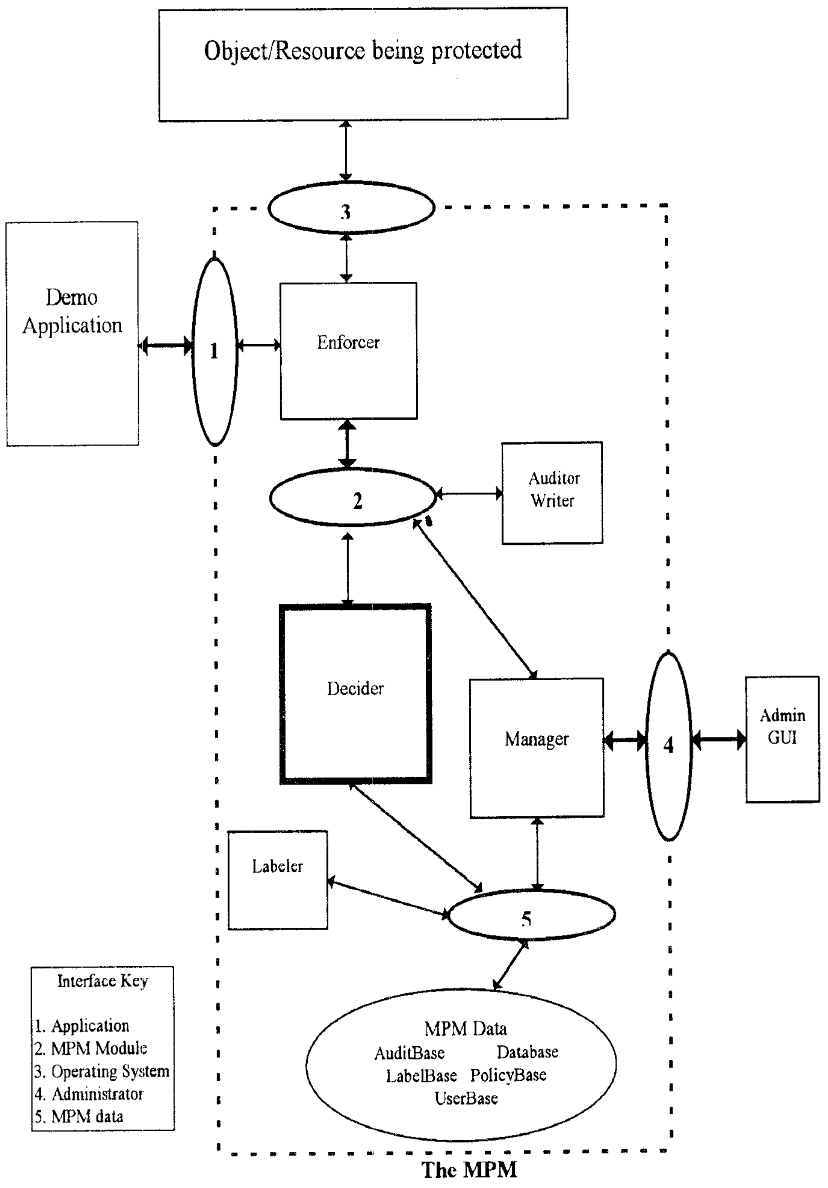

Figure 2 Preliminary Highlevel Drawing of MPM Interfaces Copyright @ 1993 Data Security, Inc.

The MPM-AT was prototyped as a "classic" database data entry program with a relatively "flat" one screen per class of data window architecture. This simplistic approach was chosen primarily to save time so the "more important" decision and enforcement modules could be given more development time. As the project continued it became painfully clear that this approach was wholly inadequate for a "production" system and that several new paradigms would be required for complex PolicyBase administration and maintenance.

Because the project goal was that of rapid-prototyping focused on the MPM-DE, I had little time to give thought to the control and enforcement of naming conventions for neither policies and domains, nor for managing their relationship (hierarchical, or otherwise) or data contents. As the MPM-DE modules matured they began to meld the polices in the PolicyBase, arriving at unexpected decisions. Upon examination it was found that, in some cases, a policy had been entered when a metapolicy should have been in place. [3] In others, the wrong policy had been entered in a metapolicy, and in still other cases the wrong rule, set of rules, or rule interaction had been specified in a policy. This problem was more of a nuisance than a project crippling occurrence, but the discovery of this situation was the beginning of understanding the complexity of administration in a multiple policy/domain environment.

Demonstrating the MPM's functionality required several sample scenarios described by various policies and metapolices. The growth of the required PolicyBase pushed the simple one-data-entry-window per class-of-data to its limits. This caused the realization that the open-ended complexity of policy specification required a supporting administration interface with literally infinite capabilities. The simple single-data-entry window per object class, and bounded internal array of objects approach, I found to be clumsy at best, requiring a great deal of user knowledge and attention, without being really capable of effectively handling PolicyBase administration in an adequately automated and dependable fashion.

The experience gained from developing the MPM administrator interface reflects, to a fair extent, the experience of Sibley, Michael, Baum, Li, and Wexelbat in the building and testing of the Policy Workbench[6]. The proposed approach, in effect, suggests taking elements of the Policy Workbench's functionality to its ultimate degree.

2. A Suggested Approach

2.A. Overview

A holistic approach seems, to the author, to offer the most effective way to deal with the complexities of policy interpretation and enforcement. The word holistic is used to represent a comprehensive, even aggressive, dynamic analysis and presentation of all data that has even a vague, ill-defined influence on policy interpretation, decision making based on the interpretation, and actions stemming from the decision throughout the whole of the MPM. This section is an abstract high level discussion of the concepts that are contained in and can leading to a Holistic Administration Tool (HAT).

2.B. Holistic Administration Tool

An intuitive approach to computer policy administration requires the development and use of software and hardware to creatively interface with human administrators in an intuitive, cognitive fashion. A cognitive approach to interfacing with computer system administrators is only beginning to be used in today's operational systems. [4] A Holistic Administration Tool (HAT) is much more than a Graphical User Interface (GUI). It must be capable of making the logical, and illogical, linkages between existing and intended policies comprehensible at an intuitive level, or at least present an administrator with reasonable approximations of what might happen because often the cascading of possible interactions is beyond immediate human comprehension, and often beyond current computer capability as well.

The implementation of a HAT, at a minimum, requires an Experienced Based (EB) system to guide an administrator through scenarios of enforcement as policies are added to the system's PolicyBase. [5] [6] A HAT should dynamically present the administrator with displays showing what information is required to create, modify, update, or delete a policy and simulates the effects of the planned changes on policy enforcement. Interactive simulation of possible scenarios is crucial to give the administrator feedback about the effectiveness of a policy and help move the

[3] A metapolicy is a "policy about policies" [11]. The MPM PolicyBase contained metapolicies that specified a conflict resolution method for the policies being enforced for a particular transaction.

[4]. A primative, relative to a full capability HAI, example is Hewlett Packard's Network Node Manager[12]. It automatically creates a map of the network when started and while up it presents the administrator with alerts by changing node colors and icons to show problem nodes and dynamically tracks nodes coming and going. This dynamically modified network map display helps the administrator gain an intuitive feel for the health of the network.

[5] Experienced Base (EB) processing is a form of artificial intelligence that limits itself to what is proven to work. It represents an engineering, rather than a theoretical approach to automated decision and data processing. A system using algorithms based on experience, rather than a generalized AI approach. This represents a trade-off of a more limited approach using what is known to work, versus a less certain, more open-ended, solve-everything-approach (personal communication with Mr. William Foster, an independent software engineer).

[6]. PolicyBase (PB) - The data used to specify, maintain, and enforce policies. The term PolicyBase was coined as a refinement of the more generalize term KnowledgeBase. The PolicyBase would be a portion of a system's over-all KnowledgeBase (KB), which, in turn, is physically stored in the system's Database (DB), an SQL based DBMS managing a RAID-5 disk farm, as an example.

administrator toward a gestalt feeling for the efficacy of the policies being implemented. The HAT should provide predictions about interactions with other policy elements as new elements are created and activated, with these predictions being incorporated into the simulated scenarios as the administrator requests. To the extent possible, the HAT will verify the effectiveness and integrity of the PolicyBase being created, but request human intervention when prediction becomes impossible or unreliable.

The impact of interactions of administrators with mature HAT tools will allow administrators to intuitively sense whether the PolicyBase will insure that decision and enforcement tools interpret policies correctly. [7] The experienced and intuitive judgment of a human administrator will often be required either because of institutional reasons, unpublished or implicit policies or "customs" of the operating entity, or because the HAT cannot resolve conflicts because of the vagueness of policy specification and/or the complexity of policy and domain interaction. The HAT may also require human input and judgment to be able to learn as the PolicyBase grows and more complex policy administration is demanded of the HAT and the administrator.

2.C. Design and Implementation Issues

Most of the basic design/implementation factors for a HAT are not unusual and are the foundation for many software and hardware tools. The over-all integrated approach and how the various factors are weighted and interact are what lifts an "everyday tool " toward the realm of a HAT. These factors include accountability, assurability, clear and specific interface defineability, evaluateability, and flexibility. This is not an exhaustive list, but will aid the reader in comprehending the inherent complexity of developing a HAT. In fact, the various factors listed may be considered policies, so the very act of attempting to design a tool or suite of tools to enforce one or more policies is, in effect, an act of policy management itself. This is one of the main reasons a HAT must, to the extent technically possible, allow the administrator to define policies and domains with as few constraints as possible, hence the flexibility requirement above. Otherwise, administrators will be reduced to enforcing the implied polices of the administration tool designers.

The actual master program may be several programs that interact. But the logical reality is that these modules will act as frameworks for the interaction of all other modules, processes, and activities that contribute to what the administrator finally sees and, more importantly, intuits. Intuit, in this context, means the administrator, through the process of policy creation and enforcement simulation, gains an insight or a feeling about the rightness or wrongness of the effectiveness of policy enforcement. In many situations there is no way to verify or guarantee that the enforcement interaction of even a simple suite of policies will result in the desired outcome in any given situation.

[7]. Intuitively based on the author's 19+ years of experience of developing and working with computer software..

One of the foundations of a HAT would be the integrity checking supplied by a commercial DBMS (Data-Based Management System). Integrity checking is a given in today's database tools and is absolutely necessary. But today's integrity checking is only a building block toward insuring the integrity of policies and rules. Holistic Integrity Assurance (HIA) will be one of the most complex and error prone services of any tool attempting to meld policies. The functionality required is not simply that of insuring data is present, but of an almost intuitive comprehension of the interaction of the data that describes a particular policy/rule and the interactions that will occur with other policies. HIA could literally be open-ended, requiring execution of all rules on all systems reachable throughout the entire world. Realistically, HIA examination and cross-checking will have to be limited to some subset of any multi-domain/network/policy universe.

Another factor to consider is that it is probable that some data will be missing during HIA checking and HAT enforcement simulation. All individual tools and tool suites must have default processing that approaches "fail safe" functionality. While baseline defaults should be hard-coded into the tools as a backup precaution, all defaults should be adjustable by an administrator. This especially means that an administrator can modify and/or completely redefine what "fail-safe" processing means in the policy universe being administrated.

The display of the effects of policy enforcement to the administrator should ideally be scenario driven. Scenario means, in this venue, simulating actual use of the data being added/modified by the administrator to do decision making and examine the consequences of actions that would have been taken if the data was active. The word data, in this context, is being used as a generic short-hand term for any element/object used to describe the policy, such as a rule, metapolicy, statement of policy goal, or specification of decision making method (quite literally any information, passive or active, associated with a policy).

2.C. I Administrator Presentation Interface

For optimum usability, the way information is presented to the administrator should be adjustable to his or her own personal preferences. While keyboard-only use is considered passé, some users might be more comfortable with this means of input, with a simple single window screen. For the simplest maintenance actives this should be allowable.

As discussed above, a much richer interface is really required to assist an administrator in arriving at/developing a gestalt understanding of the policies being entered and what enforcement decisions might be made. While a mouse, window and icon interface are likely in the near term, a virtual hood, 3-D glasses, and even a whole-body sensor stocking might be found desirable.

More over, while the optic nerve has the maximum bandwidth, the sense of smell gives the fastest feedback to the brain (the sensory nerves involved are wired directly to the brain). Other senses aggregate with senses of sight and smell to build the subconscious gestalt that many decisions are based on, which may lead to policy data some day containing odor or other sensory queues that are not consider conventional or meaningful today. Going with a holistically coined term, such an interface to an administrator might be known as a Holistic Administration Interface (HAI). By definition an HAI would provide information to an administrator via all possible sensory channels.

Unfortunately, restrictions may be necessary for technical or cost reasons and desirable for security reasons[7]. In the end, the design and implementation of the underlying PolicyBase management software may be allowed more design freedom than the design of the presentation layer that actually interfaces with the administrator.

One other issue related to a HAI that should always be kept in mind by both the designer and implementor is the nature of the default processing built into the tool and the default PolicyBase (if any) the tool starts with. If the administrator has to make too many decisions while trying to create even the simplest policy, or create data that requires data that requires yet other data to be entered and/or created, the administrator can be easily overwhelmed and either avoid using the tool when possible, or using the tool in the most minimalistic and ineffective fashion. A balance must be achieved between flexibility and restriction of the administrator's activities.

2.C.II Circular Linkages in the PolicyBase

One design problem encountered with the development of the MPM-AT was a circular dependence of system defined objects with labels which, in turn, required certain other data. The problem was handled in the MPM by allowing separate creation/entry of any interdependent data with the safety measure of checks in the decision module that resulted in a default decision and a warning to the administrator that some necessary piece of data was missing. The specific details are a function of MPM's PolicyBase data design. However, the problem itself applies to almost any attempt to express "human data" (policies in this case) in a restrictive or carefully defined form, as most modern databases do. The way humans specify policy and interpret data is often non-linear, filling in or ignoring non-existent data. Neural network processing comes closest to handling this situation. A HAT will have to have one or more mechanisms to allow the administrator to fill in data piecemeal with dependencies being ignored by the tool until the administrator says it's all there. One possible method is to always fill in defaults on the assumption that simulation feedback will remind the administrator that not all the data was supplied. Another method would be to suggest values for the missing data and guide the administrator interactively to the appropriate values. Such a feedback method might take several iterations until the administrator feels the policy, as entered in the PolicyBase, will result in the desired policy decision and enforcement. The end result would be only policies with all necessary instantiating data being enforced. However, internal checks for consistency and the presence of all necessary data is still probably a good idea (regardless of any resource cost) to provide another layer of protect from data corruption, programmer error, and malicious intent.

2.C.III Auditing

A holistic approach requires auditing for the same reasons as maintenance of databases in general. Monitoring large systems (many network nodes, many users, and/or terabyte plus databases) means either minimalistic audit trails or massive amounts of data that tell an administrator too little or too much. The implication here, again, is that one or more new paradigms will be required to benefit from human cognitive abilities to interpret and discover relationships between seemingly unrelated data. Analysis and comparison of the stored simulations, done as policies are added and modified with the actual audit results of on-going decision and enforcement tools, should aid an administrator in discovering and solving problems, and in determining which transactions should be judged as effective or ineffective as a function of the goals chosen for the policies, and whether or not the expected policies came into play for any given transaction.

2.C.IV Ownership

The ownership of a policy might be considered the issuing authority, but in many organizations local interpretations are permitted and/or different portions of a given policy may apply at different sites and they could be considered owners of their particular versions of the policy, irrespective of it being withdrawn by the issuing authority. Additionally, the implementation of a HAT may require policies to be represented by objects distributed throughout a network. With different authorities and objects present for the same original policy, ownership can become vague and the exact processing required may be affected by who is perceived to be the owner at the time. This can be a very complex issue to deal with, even for a simple maintenance rule, such as *all policies derived from a rule are withdrawn when the original issuer withdraws the parent policy.* Deciding what could actually be discarded is non-trivial because any policies derived from the parent(s) being withdrawn may themselves have become parents of other policies and may have become interwoven into policies elsewhere in the network. Ownership identification processing might help deal with the situation but unforeseen interactions resulting from the deletion/deactivation process might still occur no matter how carefully the tool design was done, the implementation crafted, and the system integration done

The above highlights the fact that no area of functionality dealing with multiple policy administration (or decision making) is really trivial. Very complex tools that interact with each other, the data being administered and used, and with the administrators and users *themselves* are required to have a truly effective and functional HAT.

2.C.V Deletion/Deactivation affects

To handle the various interactions that might occur from the deletion or deactivation of policies (such as the ownership problem, discussed above) it might be necessary to have each policy as an independent, persistent object that stays in existence (i.e., is not purged from the PolicyBase) until the HAT can be certain that no references are, or might be, made to the deleted/deactivated policy. This, in turn, implies that the delete management portion of a HAT may be as, or more, complex than the new addition and/or modification management portions of the HAT.

2.C.VI Security

In theory the HAT should be run on its own, separate, system isolated from all other systems. Practically, it may have to share a system/node with other software, which would mean that some sort of very trusted boundary protection software and/or hardware would be required to protect both the PolicyBase and all programs comprising the HAT. Since, in most cases the HAT would be aiding in the administration of one or more networks, some sort of firewall concept might be employed. Presumably the PolicyBase could be modified only by a process running on the authorized system, and the only process that could actually modify it would be the HAT. All other access by process on the same system or across a network would be via read-only access.

The HAT might also include a self-analysis capability that would provide some form of self-protection. As one example, a HAT could have internal consistency checking modules that might reside in firmware (Read Only Memory) that would watch for invalid or illegally changed data in the PolicyBase or in the HAT, itself. One example of a basic "framework" suite of policies is the DOD Goal Architecture[8]. This "base" policy suite might be kept in firmware for comparison/melding with policies placed in the PolicyBase. Additionally, as a performance and security measure most, if not all, of the modules comprising the HAT could also be in firmware, with all of the PolicyBase (or only the user defined part) residing in a changeable medium.

Another critical security concern is the fact that an administrator authorized to add and modify policies needs wide reaching, if not completely unbridled, authority to be effective. Given that the human link in the processing chain is often the weakest, it might be wise to have a separate administrator, perhaps even a different department, review the changes and apply them to the PolicyBase. This administrative application of a two-man rule would, in effect, add a quality control step, as well as a security step, to the administrative process.

Lastly, any auditing of the HAT or HAI would be directed to a non-perishable medium, such as a Write Once, Read Many times laser disk, in a limited access room not accessible by the policy administrators. Audit processing should be the least configurable feature of any security tool suite. Ideally, the audit process should never be turned off or have its output redirected, but being this draconian can interfere with actual system operation to the point of making the system impossible to use.

2.C. VII POLICY DESCRIPTIVE LANGUAGE (PDL)

The implementation of a HAT would be aided by the development and use of a Policy Descriptive Language (PDL) which would be a logical descriptive language optimized for policy description. This language would allow policies to be specified in a fashion that is much more "understandable" to the human mind and aid in the intuitive approach that is the foundation of a HAT's usability and effectiveness. This language would primarily be used by developers of a HAT, but could also be used by developers of independent policy modules that might be used by a HAT. In fact a PDL might become a standard which could lead to an industry of standard and custom policy modules that could be used on any platform that had the necessary interpretation software (residing, for example, in a HAT). A carefully crafted PDL might also aid in evaluation and certification of a HAT because once the interpretation software is evaluated as trustworthy a given PDL construct would result in a given action (decision or enforcement result) within a HAT (or one of its modules). Naturally, the argument that interactions of trusted objects may aggregate to an untrusted result, must be addressed. In fact, this uncertainty is one of the reasons the author feels a HAT with human intuitive interaction and intervention is a necessary tool for policy management.

Additionally, the implementation of a PDL interpreter/compiler would encompass the mathematical expressions necessary to describe both the "human natural" and non-linear interactions of data that effects policies.

The work of Michael [13] and Moffett [14], among others, represents a laying of the necessary foundation for a logical descriptive language that could be used to express both the explicit and implicit elements of policies, regardless of the complexity of interaction.

The development of a useful PDL (Policy Descriptive Language. I built a first-cut visual PDL with my screens.) would probably be constructed so it could express any of several policy models, as described in the work of David Bell [9] and John Dobson [10], among others.

3. Costs Versus Benefits

To actually arrive at a HAT that gives reasonably good feedback would take two to four major prototyping and test cycles. But even a first generation HAT should make policy administration more manageable and reliable. A third or fourth generation HAT could well be a Virtual Reality Interface (VRI) with an administrator literally working in a virtual world representing a complex meld of scenarios being managed by the policies being administrated. Additionally, there will probably be a playback in increased administrator efficiency[11], leading to fewer policy violations, the cost of which could, quite literally, be the existence of the enterprise for which the policies are being enforced. The improved efficiency might well pay for the expense of a sophisticated HAT of whatever generation.

Another area of potential pay-back might occur if standards were to be adopted for the policy description (perhaps expressed in a PDL) storage and processing, so that a building block approach could be used. As products using the standards matured, libraries of policies (data, enforcement, and decision modules and tools) might be built up and sold by third party vendors, resulting in greater functionality at lower cost to system administration budgets.

The cost of using a HAT would be the cost of the package, which for the early versions would probably be expensive, with additional computer capacity in speed and storage and greater cpu loading. Additionally, a major investment of time and money would be required to train administrators *and* allow them time to use the tool and make mistakes with it. The integration of a HAT into systems operations at any site would be relatively slow and probably painfully difficult.

The management and administrators of an enterprise must carefully weigh the benefits versus the losses in choosing to buy and use a HAT. The author feels that in situations where many complex, possibly conflicting, policies must come in to play to affect system activities there is no real option. Something equivalent to a HAT will have to be used, or only a few, relatively simple, policies will be manageable.

4. Conclusions

It is felt that the experience garnered from the MPM prototyping project could easily be the foundation, although a very limited, preliminary one, for the learning and understanding that comprises the "critical mass" of knowledge of design that allows the implementation of EB algorithms leading to EB modules, aggregating to EB administration tools that could be on the market within the next few years.

Research into, and commercial implementation of, a holistic approach to computer administration of policy enforcement is required today, not just in the near future. Interaction of policies from many different political entities and cultures is becoming unmanageable and, therefore, unpredictable. The increasing number and density of network connections around the world exacerbates the situation. More countries, organizations, and individuals are attempting to transact business or exchange information than ever before in human history, and it is being done between people, which often means between systems that are governed by policies with completely differing (and even totally unrelated) goals. Until recent times a human somewhere in the loop might barely manage to administer policy information by hand and easily make decisions as necessary. Custom software or hardware could be created as needed to handle specific situations, but now the flood of information and the truly immense numbers of decisions that must constantly be made are forcing shorter and shorter decision cycles, with any delay costing money or endangering lives and/or national security. Tools that take advantage of the cognitive ability of humans to filter data and extract, and often correct, decisions from a flood of data are required. Sophisticated administration tools are necessary to allow the required decision making tools to operate effectively. Effective PolicyBase administration is the foundation that any decision tool requires. As policies to be enforced become more sophisticated a HAT would, by necessity, have to become even more complex, perhaps, in time, leading system administrators into the worlds of Virtual Reality and Cyberspace.

Without a HAT, or a suite of management tools with equivalent capability, a system administrator will, at best, be able to manage a few simple policies that conflict in a relatively simple and well understood fashion. Attempting to apply and enforce more complex policies without a HAT will sooner or later lead to system failure or "policy leakage" (failure to manage the system or data as directed by the application).

As one reviewer pointed out, "The lack of such tools is a significant problem. Until it is solved, organizations are likely to implement simpler policies that either overly constrain users or fail to constrain them adequately"[8].

A great deal of work needs to be done in all the areas discussed by this paper. Given the explosion of information and the increasing complexity of human civilization (leading to more complex and interacting policies) any money expended on the research and development of such tools is money well spent.

Acknowledgments

I would like to thank David Bell and John Dobson for discussing various issues with me during the course of the MPM project; Grace Hammonds for her guidance and help; Hilary Hosmer for her support and encouragement, as well as her technical input; Liz Smith for her editorial assistance; and the sponsors of The MultiPolicy Machine research project.

This work builds upon and uses with permission work previously done for Data Security, Inc. under Air Force SBIR contract F 19628-93-C-0022.

References

[1] Miller, Arthur, The Assault on Privacy, The University of Michigan Press, 1971

[2] Schwartau, Winn, Information Warfare, Thunder's Mouth Press, New York, 1994

[3] Lewis, Ted G., "Where is Computing Headed", IEEE Computer, Vol. 27, No. 8, pp 59-63, August 1994.

[4] Hosmer, Bell, Ford, Hammonds, Nelson, Ovchinnikov, Sibert, "The MultiPolicy Machine: A New Paradigm for Trusted Systems", SBIR Phase II Final Technical Report, March, 1995.

[5] Shindyalov, I and Boure, Philip, "The Shape of Databases To Come", DEC Professional, Vol. 11, No. 12, pp 38-43.

[6] Sibley, Michael, and Wexelblat, "Use of an Experimental Policy Workbench: Description and Preliminary Results", Proceedings of the IFIP TC11.3 5th Working Conference on Database Security. North-Holland, Amsterdam, 1991.

[7] Rooker, Terry, "Application Level Security Using an Object-Oriented Graphical User Interface", ACM *SIGSAC New Security Paradigms Workshop*, pp 105-117, August, 1992-1993.

[8] DISA/CISS, "Department of Defense Goal Security Architecture", draft of 1 August, 1993.

[9] Bell, David, "Modeling the 'MultiPolicy Machine'", ACM SIGSAC New Security Paradigms Workshop, August, 1994.

[10] Dobson, John, "A Framework For Expressing Models Of Security Policy", Proceedings of the 1989 IEEE Computer Symposium on Security and Privacy", pp 229-239, May, 1989

[11] Nielsen, Jakob, "Iterative User-Interface Design", IEEE Computer, Vol. 26, No. 11, pp 32-41, November 1993.

[12] Hosmer, Hilary, "Metapolicies II", Proceedings of the 15th National Computer Security Conference, October, 1992.

[13] Michael, Sibley, Baum, and Li, "On the Axiomatization of Security Policy: Some Tenative Observations about Logic Representation", Submitted to the Sixth IFI WG 11.3 Working Conference on Database Security.

[14] Moffet, Jonathan, and Sloman, Morris , "The Representation of Policies as System Objects", Proceedings of Conference on Organizational Computer Systems. SIGOIS Bulletin vol 12, nos 2 & 3 pp 171-184, 1991

Bibliography

Gasser,M. Building A Secure Computer System, Van Nostrand Reinhold, 1988.

Hoc, Green, Samurcay, and Gilmore (Editors),Psychology of Programming,Academic Press, 1990

Hosmer, Hilary, "The Multipolicy Paradigm for Trusted Systems", Proceedings of the New Security Paradigms Workshop, Little Compton, R.I., 1992, IEEE Press 1993

Hosmer,Bell,Hammonds,Nelson,Ovchinnikov, "The MultiPolicy Machine: A New Paradigm for Trusted Systems", SBIR Phase II Interim Technical Report, March, 1994.

Neugent, Bill and Burgoon, Mike and Firey, Jeanne and Rudell, Mindy, "Modern MultiLevel Security (MLS): Practical Approaches for Integration, Certification, and Accreditation", NIST/NCSC Proceedings of the 17th National Computer Security Conference, Volume 1, pp 309-317,

[8] Anonymous reviewer.

October, 1994.

Sandhu, Ravi, "Lattice-Based Access Control Models", IEEE Computer, Vol. 26, No. 11, pp 9-19, November 1993.

Sibley, Michael, and Wexelblat, "An Approach To Formalizing Policy Management", Proceedings of the 2nd International Conference on Economics and Artificial Intelligence. Pergamon Press, Oxford, England, 1991.

Vazquez-Gomez, "Modeling Multidomain Security", ACM SIGSAC New Security Paradigms Workshop, pp 167-174, August, 1992-1993.

[WD] Webster's Unabridged Dictionary of the English Language, Portland House, 1989.