To be published in IEEE *IT Pro Magazine* in the Perspectives Column, a non-refereed section of the magazine

A Matter of Policy

David Ferraiolo, Jeffrey Voas

US National Institute of Standards and Technology

George Hurlburt

Change-Index Inc.

Today's computationally intensive systems are assemblages of components that are composed to offer specific functions. The Naval Air Systems Command once defined a system as a "collection of hardware, software, people, policies and procedures, working together as a whole, under regulated conditions". This definition was particularly interesting for aircraft, not necessarily computational resources, since aircraft were the "ultimate" system product line in terms of complexity. However the current definition of Public Key Infrastructure (PKI) embraces the "set of hardware, software, people, policies and procedures needed to create, manage, distribute, use, store, and revoke digital certificates". Interestingly, policy plays an explicit role in both of these definitions where software is also an indispensable component.

Policy frequently intermingles with the hardware, software, people and procedural system components across a broad spectrum of systemic thinking. This includes not only aircraft and PKI, but also extends to: geographic information systems, medical records, Internet security, and many other systemically related arenas. Despite these associations, the effect of implementing policies on system behaviors remains more art than science. The literature is less than replete with concrete system policy implementation guidance.

To many, system policy is a statement posted on a website indicating intention to protect personal data. In reality, however, policy is much broader and its enforcement far more consequential.

Information system security policies are a set of rules that when implemented afford a strategy for the protection of information. The policy objectives are diverse and span the social-economic spectrum. System policies govern such elements as individual privacy, selective access to proprietary information, national security protection, fraud prevention, data surety and conflict-of-interest. Policy can derive from laws and regulation, but may also stem from business culture and its tolerance for risk. Correspondingly, policies also vary considerably from one institution to another. The policies of a hospital will differ dramatically from those of a financial institution, or that of a military agency. Further, policies are often driven by laws specific to regulated vertical markets.

Unfortunately the state of practice regarding policy enforcement is weak, even when procedures are factored into the preventative mix. For perspective, consider a typical computing environment, including the several used to create this article, where all information that can be read can also easily be leaked to anyone in the world. The enforcement of system policy often depends on procedures that are imposed on computer users. This may include statements like “Do not put patient data on memory sticks, laptops, or in email”, “delete credit card information after processing a transaction” or "flash memory is not allowed in this organization". The problem here is that procedures break down, users become complacent, or human errors occur. Perhaps the most problematic issue is that the very users that are instrumental in the enforcement of policy often possess the highest motivation to violate established policies. Consider the US$7.2 billion fraud case at Société Générale caused by a 31-year-old rogue trader, who was able to bypass internal control procedures. This underscores just how critical adequate operational risk management and policy enforcement have become to trustworthy banking practices.

In the early days of shared computing, policy issues mainly pertained to who could read and write what files - all within the confines of a single and largely isolated system. Since that period, however, computing has become increasingly distributed, and applications have simultaneously become sophisticated and interdependent. Today, policies need to be enforced within and across a multitude of heterogeneous file management systems, and across such applications as email, workflow and records management. Associated with these systems and applications are specific operations and resource types over which policies need to be enforced. As such, policy enforcement today needs to contend with a large variety of operations to include read/write,

send, review, approve, insert, and copy/cut-paste. Worse, these operation types are applied to a large variety of resource types like files, messages, attachments, work items, records, fields and clipboards. Making matters still worse, these operations are performed under the control of a multitude of systems and applications often running simultaneously and with great interdependence. These interdependencies enable emergent behaviors, and when these behaviors are deemed undesirable, it becomes the proverbial “he said” “she said” problem, with fingers pointed in all directions when determining the root cause(s).

What if policy-derived rule sets could be rigorously defined and automated for software intensive systems? Imagine a “Policy Machine” that allows codification of arbitrary rules stemming from policy to create executable code. When enabled, this code constrains the system to behaviors that comply with declared policies. Moreover, policy-derived executables may also be targeted to constrain undesirable behavior at the middleware or applications levels. Imagine that the “Policy Machine” is capable of highlighting potentially conflicting policy-derived rules for human resolution. Such a tool exists today at the US National Institute of Standards and Technology (NIST), the NIST Policy Machine [1]. It offers a new technology in enforcing the important role of policy in systems design, evolution, management, and policy enforcement.

The NIST Policy Machine is a step towards the idea of having an “Enterprise Operating System”. It provides a means of translating strategic policy into executable rules, thus satisfying a requirement to enforce tactical procedures that enforce the policy. In this instance, however, human operators are not required for systemic rule enforcement. The importance of the NIST Policy Machine is significant in light of emergent computational capabilities.

The cloud, which features dynamically composed services, presents opportunities for the introduction of data security concerns because dynamic binding of any type of composed service or functionality can foster undesirable emergent behaviors. Seemingly innocuous new code that offers additional services, as well as applications and their associated data, can easily harbor veiled malicious payloads. As open services proliferate throughout cloud servers, their numbers will outstrip human inspection capacity. Hence some form of automated inspection will be required to curtail the introduction of surreptitious malware. Moreover, as malware morphs in character, prevailing policies must morph accordingly. Thus, a Policy Machine oriented around restricted user permissions and perhaps augmented by some form of intelligent pattern recognition could be brought to bear within the cloud environment to address service insecurity. The same principle holds true for open mobile applications which are proliferating at staggering rates, especially in the Android community. In this instance, a Policy Machine “Lite” could be enacted at the “app” level to: (1) provide policies that block apps with malicious payloads, or (2) block promiscuous data dissemination practices contrary to good privacy protection practices.

The Policy Machine manages access by enforcing policy over application specific operations and object types. Many application specific operations and their services can be accommodated as sequences of Policy Machine administrative operations. These operations distribute/revoke capabilities to/from users to perform file level operations on objects. For example, e-mail services can be provided through the distribution of read operations on files that are treated as messages and attachments. Likewise, workflow management services can be provided through

the distribution of capabilities such as read/write operations permitted to a prescribed sequence of users operating on files that are treated as work items. Consequently, a large variety of application specific object types can be treated generically as Policy Machine objects, and therefore protected under the Policy Machine. Of further significance is that the Policy Machine not only enforces policy over operations on heterogeneous objects; it actually provides an environment where Policy Machine native features reduce or eliminate the need for application-level access control code [1].

With a few exceptions, privilege management systems represent the current state of the practice in enforcing policy. Privileges specify and control which users can perform which operations on which resources. These systems come in many forms, with an emphases placed on efficiency, intuition, and visualization for the management of potentially millions of privileges.

While representation of privileges is fundamental to the expression of policy, it is not sufficient in the support of all policies. For instance, management of privileges alone are insufficient in preventing the “leakage” of data content to unauthorized principals, whether through Trojan horse attacks, or malicious or complacent user actions. Confinement is a security property that prevents the leakage of data content outside a designated set of resources. For instance, confinement rules are critical in the enforcement of policies such as “only doctors can read the content of medical records”, and “only my designated closest friends can read my personal data”.

Users ability to trust the integrity of their data is a fundamental goal of creating policies. One way to preserve the integrity of protected data is to limit access to it. Such policy, however, requires some *a priori* knowledge of the user's role in the organization. Together, roles and rules combine to determine the specific conditions, under which individual users may access, manage or share sensitive information. These provisions extend to specific users based upon predetermined user profiling, but must also abstract to embrace the hierarchal chain of command. The NIST Policy Machine addresses this – it permits rules that can accommodate roles either at the personal or the organizational level. In this manner, policies may also dictate complex controls. To perform an operation on a resource, policy may prescribe, for example that a user: (1) has a “need-to-know”, (2) is appropriately cleared, (3) is competent, (4) has not performed a different operation on the same object, (5) is incapable of accessing other enterprise objects, or (5) is only capable of accessing an object or any copy of the object while in performance of a specific task.

Some policies are not only concerned with ensuring authorized access, but are also concerned with knowing who has access to what sensitive information. Under these policies, the “owner” of a resource, or a fiduciary, imposes the requirement to track access. Access tracking is a concern of many data control and privacy policies, i.e., “I know who can currently read my data or personal information, and I can revoke that access”. Because information objects can be renamed, copied, and given away, tracking the dissemination and retention of access is difficult or impossible through privilege expressions alone. The Policy Machine rule base accommodates such dynamic individualized access controls.

Circumstantial separation of capabilities defines a set of access events (circumstances) that if encountered result in “establishment conditions” that deny users certain access privileges. Among other policy objectives, circumstantial separation of capabilities is instrumental in enforcing separation of duty and conflict-of-interest policies. Separation of duty is a security principle used to formulate multi-person control policies, to reduce the likelihood of the occurrence of fraud, by requiring that two or more different people be responsible for the completion of a sensitive task or set of related tasks. A simple example would be a user capable of both requesting and approving a purchase order. The effective rule would prohibit a user who requests a purchase order from approving the same purchase order.

Circumstantial separation of capabilities is also instrumental in enforcement of Chinese wall policies. Chinese wall policies are designed to address conflict-of-interest issues. When a user gains access, for example to the competitive practices of two competing entities, the user gains knowledge amounting to insider information. This information can undermine the competitive advantage of one or both entities or can be used for personal profit. To address these issues two sets of data must first be defined. A circumstantial separation of capabilities rule can then be invoked to prohibit a user who reads data in one set from reading data in the second. Like many other policies, Chinese walls depend on multiple properties. To prevent a conflict-of-interest situation through a Trojan horse attack, applied confinement rules would be required as well.

The Policy Machine attempts to solve the global enforcement problem by offering a type of “universal operating environment” that allows the same operational effect as many conventional systems and applications, all under the control of the Policy Machine. The approach is to treat all protected objects agnostically and treat all controlled operations, extending to privilege management, as Policy Machine recognized operations and/or as abstractions on that set of operations. Specifically the Policy Machine recognizes two sets of operations: (1) create, read, write, and delete for non-administrative objects, and (2) its own set of administrative operations for administrative data. The first set of operations naturally applies to applications that treat objects as files - such as, word processors, text editors, and drawing packages.

To demonstrate the Policy Machine’s viability, NIST has developed a reference implementation and can now demonstrate the expression and enforcement of a wide variety of policies to include those exhibiting confinement, access tracking and circumstantial separation of capabilities. (This reference implementation is based on the architecture described in [1].) In addition, NIST can demonstrate comprehensive enforcement of policy over a rich user environment. This environment includes the Open Office suite of applications, email, workflow, forms, and records management, as well as forms of inter-process communication to include copy/cut and paste. As an indication of the Policy Machine’s comprehensiveness, consider a user that copies and pastes the content of a field of a record into the body of an email message. Under this scenario, in order to read the message, the recipient would need to be permitted to read the field of the record.

All of the above mechanisms become paramount as service-based clouds become an increasing reality. Moreover, as systems fold into networked federations representing system-

of-systems, the degree of complexity increases well beyond the abilities of having procedurally savvy people in the system loop. It becomes too difficult to manually monitor and control system activity. System-of-systems concurrency establishes circumstances where a rule fired in a given system precipitates another rule to be fired in a related downstream system. In an instant, however, based upon changing environmental conditions, the same rule in the initial system may invoke a completely different set of rules in the downstream system. For example, a user can attach a record, created under a relational database management service, to an email message and send that record to anyone of his/her choosing, however regardless of the recipient of the message, any user that opens the attached record would only be able to read and/or write fields for which he/she is authorized. If the permissions change on the record, the capabilities for recipients to perform operations on the attachment change in lock-step.

The ability to employ meta-rules becomes important to curtail the possibility of contradictory or potentially chaotic rule behavior generated by individual systems subject to intrinsic rules of their own. In a deeper sense, meta-rules are also necessary to manage user concurrency at one level of the organization, where behavior may appear messy and unfocused at the working level, while higher echelons still need a unified picture of organizational behaviors and trends irrespective of the details. Nonetheless, as exceptions are encountered, selective “drill-down’ becomes necessary to isolate the patterns that spawn exceptions. The same principle, over time, may well constitute system-of-systems cyber-security thinking, where anomalous usage patterns emerge. By defining policies that seek patterns that appear to violate system integrity, novel attacks can be identified and analyzed well before they are fully defined or understood.

Expressing and enforcing advanced policies across applications and systems is difficult to impossible with existing privilege management mechanisms. The Policy Machine rectifies this shortfall. A logical “machine” comprising a fixed set of data relations, the Policy Machine is configurable through a fixed set of administrative operations. These operations permit the expression of combinations of many policies, and a fixed set of functions for making access control decisions, and ultimately, enforcing policy based on that expression. Once fully matured, it’s potential to grow into a logic driven, full-blown configurable “Enterprise Operating System” is real.

References

[1] D. Ferraiolo, V. Atluri, and S. Gavrila, “The Policy Machine: A Novel architecture and framework for access control policy specification and enforcement”, *J. of Systems Architecture*, Elsevier, (2010).

For more information on the NIST Policy Machine, contact Dave Ferrariolo at NIST at dave.ferrariolo@nist.gov.