A Matter of Policy

David Ferraiolo and Jeffrey Voas, *US National Institute of Standards and Technology*
George F. Hurlburt, *Change Index*

Today's computationally intensive systems are assemblages of components that offer specific functions. The Naval Air Systems Command (NASC) currently defines a system as "An aggregation of end products and enabling products to achieve a given purpose."[1] NASC 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 as aircraft, not computational resources, comprised the ultimate systemic product line. Interestingly, 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."[2] Interestingly, policy plays an explicit role in both of these definitions, where software is 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 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.

Current System Policies

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 integrity, and conflicts of interest.

Policy can derive from laws and regulation but can also stem from business culture and its tolerance for risk. Policies thus vary considerably from one institution to another. The policies of a hospital will differ dramatically from those of a financial institution or military agency. Furthermore, policies are often driven by laws specific to regulated vertical markets.

Weak Enforcement

Unfortunately, the state of practice regarding policy enforcement is weak, even when procedures are factored into the mix of security governing components. For perspective, consider a typical computing environment, 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 might include statements such as, "Don't put patient data on memory sticks or laptops or in emails," "delete credit card information after processing a transaction," or "flash memory isn't 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 who are instrumental in policy enforcement 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.[3] This underscores just how critical adequate operational risk management and policy enforcement have become to trustworthy banking practices.

Growing Complexity

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 must be enforced within and across a multitude of heterogeneous file management systems and across email, workflow, and records-management applications. Associated with these systems and applications are specific operations and resource types over which policies must be enforced. As such, policy enforcement today must contend with a large variety of operations, including read-write, send, review, approve, insert, and copy- or cut-and-paste operations.

Worse, these operation types are applied to a large variety of resource types such as files, messages, attachments, work items, records, fields, and clipboards. Furthermore, 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.

A Policy Machine

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 could be targeted to constrain undesirable behavior at the middleware or application levels.

Imagine that the policy machine can highlight potentially conflicting policy-derived rules for human resolution. Such a tool exists today at the US National Institute of Standards and Technology.[1] The NIST Policy Machine offers a new technology in enforcing the important role of policy in systems design, evolution, management, and policy enforcement.

Policy Enforcement

The NIST Policy Machine is a step toward an "enterprise operating system." It provides a means of translating strategic policy into executable rules, thus satisfying a requirement to enforce tactical procedures that in turn enforce the policy. In this instance, however, human operators aren't required for systemic rule enforcement. The importance of the NIST Policy Machine is significant in light of emergent computational capabilities.

Access Control

The NIST Policy Machine manages access by enforcing policy over application-specific operations and object types. It can accommodate many application-specific operations and their services as sequences of Policy Machine administrative operations that distribute and revoke capabilities to and from users on file-level operations. For example, it can provide email services through the distribution of read operations on files that it treats 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, the Policy Machine not only enforces policy over operations on heterogeneous objects but also provides an environment where its native features reduce or eliminate the need for application-level access control code.[4]

Privilege Management

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 emphasis placed on efficiency, intuition, and visualization for the management of potentially millions of privileges.

Although representation of privileges is fundamental to the expression of policy, it's not sufficient to support all policies. For example, management of privileges alone is insufficient in preventing the "leakage" of data content to unauthorized principals, whether through Trojan horse attacks or through malicious or complacent user actions. Confinement is a security property that prevents the leakage of data content outside a designated set of resources. For example, confinement rules are critical to enforcing policies such as "only doctors can read the content of medical records" or "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 data integrity 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 can access, manage, or share sensitive information. These provisions extend to specific users based on predetermined user profiling, but they must also abstract to embrace the hierarchal chain of command.

The NIST Policy Machine addresses this by permitting rules that can accommodate roles at either the personal or organizational level. In this manner, policies might also dictate complex controls. To perform an operation on a resource, policy can prescribe, for example, that a user has a "need" to know, is appropriately cleared, is competent, hasn't performed a different operation on the same object, is incapable of accessing other enterprise objects, or can only access an object or any copy of the object while performing a specific task.

Access Tracking

Some policies are concerned not only with ensuring authorized access but also 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—that is, "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

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 multiperson control policies to reduce the likelihood of fraud by requiring that two or more people be responsible for the completion of a sensitive task or set of related tasks. So, for example, a user who requests a purchase order wouldn't be allowed to approve the same purchase order.

Circumstantial separation of capabilities is also instrumental in enforcing "Chinese wall" policies, which are designed to address conflict-of-interest issues. When a user gains access, for example, to the competitive practices of two competing entities, he or she gains knowledge amounting to insider information. This could undermine the competitive advantage of one or both entities or could be used for personal profit. Addressing these issues requires defining two sets of data. 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 set. 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.

Global Enforcement

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 Policy Machine control. The approach is to treat all protected objects agnostically and treat all controlled operations, extending to privilege management, as Policy-Machine-recognized operations or as abstractions on that set of operations.

Specifically, the Policy Machine recognizes two sets of operations: first, create, read, write, and delete for non-administrative objects; second, 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.

Implementation

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 an architecture described elsewhere.[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 interprocess communication to include copy or cut and paste. As an indication of the Policy Machine's comprehensiveness, consider a user who copies and pastes the content of a field of a record into the body of an email message. Under this scenario, to read the message, the recipient would need to be permitted to read the field of the record.

All of these mechanisms become paramount in service-based clouds. Moreover, as systems fold into networked federations representing a system of systems, the degree of complexity increases well beyond the capabilities of 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 on changing environmental conditions, the same rule in the initial system can 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 or her choosing. However, regardless of the message recipient, any user who opens the attached record would only be able to read or write fields for which he or 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 metarules 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, metarules are also necessary to manage user concurrency at one level of the organization, where behavior might 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, might well constitute system-of-systems cybersecurity 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're fully defined or understood.

Expressing and enforcing advanced policies across applications and systems is difficult if not 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, its potential to grow into a logic driven, full-blown configurable "enterprise operating system" is real.

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

References

1. J. Martin et al., "Systems Engineering Guide," Naval Air Systems Command, May 2003.
2. A. Valjarevic and H.S. Venter, "Towards a Digital Forensic Readiness Framework for Public Key Infrastructure Systems," *Proc. Information Security South Africa* (ISSA 11), IEEE Press, 2011, pp. 1–10.
3. C. Matlack, "Société Générale's Fraud: What Now?" *Bloomberg Business Week*, 24 Jan. 2008; www.businessweek.com/globalbiz/content/jan2008/gb20080124_769729.htm.
4. D. Ferraiolo, V. Atluri, and S. Gavrila, "The Policy Machine: A Novel Architecture and Framework for Access Control Policy Specification and Enforcement," *J. Systems Architecture*, vol. 57, no. 4, 2011, pp. 412–424.

David Ferraiolo *is a group manager at the US National Institute of Standards and Technology (NIST). His research interests include computer security and information policy. Contact him at dave.ferrariolo@nist.gov.*

Jeffrey Voas *is a computer scientist at the US National Institute of Standards and Technology (NIST). He also serves as IEEE Division VI Director. Voas is a Fellow of IEEE and AAAS. Contact him at j.voas@ieee.org.*

George F. Hurlburt*, CEO of Change Index, has an abiding interest in applied complexity analysis. He retired with a Meritorious Civilian Service Award after 38 years as a Navy Senior Systems Analyst, where he pioneered collaborative network architectures for the US Department of Defense Test and Evaluation Community. Contact him at ghurlburt@change-index.com.*