On Policies and Intents*

Matthew L. Bolton, Celeste M. Wallace, and Lenore D. Zuck

University of Illinois at Chicago
{mbolton,cwallac2}@uic.edu, lenore@cs.uic.edu

Abstract. A policy is a set of guidelines meant to accomplish some intent. In information security, a policy will take the form of an *access control policy* that describes the conditions under which entities can perform actions on data objects. Further, such policies are prolific in modern society, where information must flow between different enterprises, states, and countries, all of which will likely have different policies. Unfortunately, policies have proven to be extremely difficult to evaluate. Even with formal policies, basic questions about policy completeness and consistency can be undecidable. These problems are confounded when multiple policies must be considered in aggregation. Even worse, many policies are merely "formal-looking" or are completely informal. Thus, they cannot be reasoned about in a formal way and it may not even be possible to reliably determine whether a given course of action is allowed. Even with all of these problems, policies face issues related to their validity. That is, to be valid, a policy should reflect the *intent* of the policy makers and it should be clear what the *consequences* are if a policy is violated. It is the contention of the authors that when evaluating policies, one needs to be able to understand and reason about the policy maker's intentions and the consequences associated with them. This paper focuses on the *intent* portion of this perspective. Unfortunately, because policy makers are humans, policy maker intentions are not readily captured by existing policy languages and notations. To rectify this, we take inspiration from task analytic methods, a set of tools and techniques human factors engineers and cognitive scientists use to represent and reason about the intentions behind human behavior. Using task analytic models as a template, we describe how policies can be represented in task-like models as hierarchies of goals and rules, with logics specifying when goals are contextually relevant and what outcomes are expected when goals are achieved. We then discuss how this framing could be used to reason about policy maker intent when evaluating policies. We further outline how this approach could be extended to facilitate reasoning about consequences. Support for legacy systems is also explored.

Keywords: Policies, Intent, Access Control, Firewalls, Complex Systems.

1 Introduction

The term "security policy," even when only considered in the information security domain, has numerous definitions. This paper is concerned with security policies that

* This research was supported in part by NSF grants CCF-1141863, CNS-1228947, IIS-0747369, and IIS-0812258.

V. Venkatakrishnan and D. Goswami (Eds.): ICISS 2012, LNCS 7671, pp. 104–118, 2012.

specify under what conditions entities can perform certain actions on data objects. Such policies are broadly referred to as *access control policies*. One can even view other policies, such as Health Insurance Portability and Accountability Act of 1996 (HIPAA), the Financial Services Modernization Act of 1999 (Gramm-Leach-Bliley Act, or just GLB), and the European Data Protection Directive 95/46/EC (EU Directive 95/46/EC 1995) as access control policies, although they are presented in natural language and are thus much harder to reason about.

All policies, whether formal or not, aim to capturing some "intent." For example, a network policy that allows for nothing but the communication of TCP packets on port 22 only permits ssh communication. Thus, the intent of allowing only ssh communication can be translated to a more formal requirement of "block all but TCP communication on port 22" which in turn can be expressed as a firewall policy or in some logics.

Policies have been the focus of a significant amount research for over four decades (e.g, [BL75]). It is well known that many decision problems based on access control policies are undecidable [HRU76]. This is not surprising since access control mechanisms are often expressed in languages that at best can be translated into first order logic, and at worst (and more commonly) into human "legalese" that has no semantics. As stated above, a policy determines the conditions under which entities are allowed to perform actions on data. Thus, we can simplify the discussion and assume that the goal of a policy is to provide a boolean response to questions regarding whether some actions are allowed in given circumstances.

The works above show that even the simplest questions, such as *completeness* – does a policy establish the access for each instance - and *consistency* - whether a policy can establish both access, and the lack of it, for the same instance –, are usually undecidable. Moreover, when access cannot be established for a given instance, or when contradictory access can be established, there are two main problems that may need to be resolved: the *intent* of the policy, and the *consequences* associated with taking actions not prescribed by the policy. Even when policies are complete and consistent (and computationally tractable), intent and consequences may play a crucial role. The former is, in a sense, harder to determine – a policy that is internally consistent and complete may still not be consistent with the intent behind it (we shall show some simple examples of this using Section 2). Yet, at times, as we argue, some reasoning about intent may allow us to detect such anomalies. Consequences naturally come up in the case of contradictory (inconsistent) policies, where the determination of whether or not one can access the forbidden (or vice versa) may depend on the consequence of the access. This often occurs when policies are merged, and one policy grants access only under certain terms, which contradict those under which access is granted in the other policy. The issue of intent also arises in the case of incomplete policies. For example, separate access may be granted to all the individual entities that shares the same attributes except for one. It may not be clear why access was not given to this individual: there could be a valid reason, or there may be an error of omission or commission. In such a situation, the benefits offered by giving the entity access may outweigh the perils of doing so.

With digital data replacing almost all of the analog data, data sharing, transfer, and dissemination have become a part of our everyday life. As is often the case with emerging technologies, decades pass before awareness of the perils of the technology

develops. In the case of data, more and more "privacy policies" have emerged to control data sharing. These can all be viewed as access control policies. Such policies do not only attempt to dictate when and how data can be shared (here, we are using the term "share" loosely) between parties within an organization, but also between organization and even nations. Policies such as HIPAA, GLB, and EU Directive 95/46/EC are not expressed in any formal framework and leave much room for interpretation. But even if they were amenable to formal treatment, basic questions of completeness and consistency are unlikely to be expressible in a decidable logic. For example, these acts contain clauses such as "unless this violates another clause in this act" which, to be expressed formally, would require a logic beyond first order.

These types of statements often lead to situations where it is virtually impossible to determine what the right course of action is since policies that are incomplete or inconsistent do not uniquely determine such. In fact, an inconsistent policy may allow for two contradictory actions. To determine the "right" course of action, it is necessary to reason about the *intent* of the policy. In fact, as is well known, even policies that are complete and consistent may not be consistent with the intent behind them, having been constructed by humans through a patchwork of numerous revisions over a long period of time where the implications between policy elements may not be completely understood. Moreover, when a policy does not uniquely determine a course of action, and one must be taken, it is necessary to consider the *consequences* of the possible actions. These consequences may not always be objectively ordered (e.g., "pay an insurance premium and maintain your reputation as a reliable helpful physician" vs "refuse to help a colleague" vs "illegally keep data and maintain your reputation, unless you are caught") and, while automation can assist in listing options, it is ultimately a human that has to decide among them.

This paper focuses on intent as a methodology to analyze, reason about, and develop policies.

We start our discussion by using a relatively simple and well studied access control policies – firewalls. Such policies are always complete and consistent, moreover, they are Effectively Propositional (EPR, also known as the Bernays-Schönfinkel class) and enjoy some other mathematical properties that make them easier to reason about than most others. Yet, as has been pointed out repeatedly in the literature, they may have some issues pertaining to intent (see, e.g., [ASH03, ALRP12, YMS$^+$06]).

We then focus on the issue of intent and its application in policies. A policy is a representation of a human policy maker's intent, thus, we examine the way that cognitive scientist and human factors engineers reason about human intentions. This examination reveals that the constructs in task analytic models that allow the intentions of human behavior to be reasoned about can also be used to reason about human policy intentions. Using this as inspiration, we adapt task analytic modeling concepts into a form that allows them to be used to express policy in terms of the policy maker's intent. We then explore how this new formulation can be used to reason about policy intentions. In taking this route, we encounter a common problem from model-based engineering. Specifically, the language or tool used to represent a concept (such as intent) will necessarily constrain how that concept can be evaluated and reasoned about. This is often the case with firewalls where intent can be circumvented by redirection of ports, and

only deep packet inspection (which, as we know, may violate net neutrality) can offer guarantees one expects to obtain from firewalls. We discuss how our proposed, more expressive, modeling concept could be used synergistically with these less expressive, legacy policy systems.

We chose to ignore numerous technical issues that are beyond the scope of this paper and of current knowledge. Such issues include how to implement (perfect) sanitization, encryption, and data destruction. Rather than arguing about their feasibility, we consider such (perhaps unattainable) requirements as simple obvious tasks and focus on the policy statement rather than on the actions it calls for.

2 Firewall Policies

In this section, we discuss the ideas of policies and intent as they apply to firewalls. We choose firewalls because of their simplicity. They are well understood policies that are always complete and consistent, have a clear semantics, and questions like completeness and consistency are decidable.

2.1 Firewall Policies

A firewall is a program that controls the flow of information into and out of a computer or network. Traditionally, most firewalls act as a barrier between a local network or a personal user and the Internet. When a file (referred to as a "packet") from the Internet is sent to the protected computer, the firewall determines if that file should be allowed or blocked. Due to the widespread amount of malicious activity on the Internet, it is essential that a user's firewall be configured properly: allowing too many packets in subjects the user to potential harm, while blocking too many packets would prevent the user from doing anything useful. As a result, much work has been done to analyze firewall policy to ensure that a given policy behaves as desired (see, e.g., [ASH03, ASH04, ALRP12]).

Firewall policies are configured as a sequence of rules that describe whether to allow or block a packet. They are generally of the form

$$\mathcal{P} : \langle p_1, a_1 \rangle; \langle p_2, a_2 \rangle; \ldots; \langle p_n, a_n \rangle$$

where for every i, $1 \leq i \leq n$:

1. p_i is a condition of the type of packet: which protocol, source IP and port, destination IP and port, and p_n is always T (true);
2. $a_i \in \{\texttt{block}, \texttt{allow}\}$.

This means that such a policy applies conditions in order, and the first one that applies determines whether the packet is allowed or blocked. More formally, let $x \models p$ denote that packet x satisfies condition p. Then, the action the policy applies to a packet x is a_i, where i is the minimal such that

$$x \models p_i \qquad \text{and for all } j < i,\ x \not\models p_j$$

Since $p_n = \mathtt{T}$, it is always the case that $x \models p_n$. Consequently, the policy guarantees that every packet is either allowed or denied.

Given a packet x and a policy $\mathcal{P}$, we say that x *is allowed by* $\mathcal{P}$ if the action taken on x per $\mathcal{P}$'s rules is `accept`, and that x *is blocked by* $\mathcal{P}$ otherwise. Let $A(\mathcal{P})$ (reps. $B(\mathcal{P})$) denote the set of packets that are allowed (reps. blocked) by $\mathcal{P}$. For every set of packets Γ there exists a policy $\mathcal{P}$ such that $\Gamma = A(\mathcal{P})$ (this is because there are finitely many packet types, and one can always enumerate the set of packets that are allowed, and block the others). We say that two policies $\mathcal{P}$ and $\mathcal{Q}$ are *equivalent*, denoted by $\mathcal{P} \equiv \mathcal{Q}$, if $A(\mathcal{P}) = A(\mathcal{Q})$.

Consider now two policies, $\mathcal{P}$ and $\mathcal{Q}$. The cascading, or chaining, of $\mathcal{P}$ and $\mathcal{Q}$, is the policy resulting by applying $\mathcal{Q}$ to the packets that are allowed by $\mathcal{P}$. Denote the packets allowed by this cascading as $A(\mathcal{P}) \rhd A(\mathcal{Q})$.

Following the observation above, there is a policy that is equivalent to $A(\mathcal{P}) \rhd A(\mathcal{Q})$. We can construct such a policy, but omit the description for brevity. See [Gut00] for examples. The following observation establishes that cascading is commutative, and that the set of packets allowed by the cascading of any two policies is the intersection of the sets allowed by each.

Observation 1. *For any policies* $\mathcal{P}$ *and* $\mathcal{Q}$*,*

$$A(\mathcal{P}) \rhd A(\mathcal{Q}) \;=\; A(\mathcal{Q}) \rhd A(\mathcal{P}) \;=\; A(\mathcal{P}) \cap A(\mathcal{Q})$$

Proof. Let x be a packet. Then from the definition of $\rhd$, $x \in A(\mathcal{P}) \rhd A(\mathcal{Q})$ iff it is allowed by $\mathcal{P}$ and then by $\mathcal{Q}$, thus, it is in both $A(\mathcal{Q})$ and $A(\mathcal{P})$. □

This observation is not, in general, true. Consider a policy that says "allow x" and one that says "block x" without the precedence assumptions. Then cascading the policies one way would allow for x, cascading in another way would block x, and the conjunction will be trivially F, making it flawed. Since, as we argue soon, cascading and intersecting policies are of great importance, it's just nice to observe that focusing on firewall policies restricts the problem in a way that allows us to ignore much of the mechanics involving operations on policies.

Cascading and chaining policies are of particular importance in the case of firewalls, where an enterprise packets may have to pass through different firewalls to get into different part of the enterprise network. Of course, if a network allows several routings for packets, the results may be conflicting – i.e., one source-to-destination routing may result in a different allow/block decision than a different routing of the same packet. While checking whether such conflicts can arise is decidable, in practice, the computation may be impractical.

The intersection of policies, however, is crucial when having to apply several policies to the same object. For example, when data "crosses" borders it may need to be allowed by all policies of the countries involved. We shall return to this point later.

For now, we focus on firewall policies because of their relative simplicity.

2.2 Policy vs. Intent

Obviously, nobody wants a policy to be inconsistent or incomplete. Such eventualities indicate that the policy does not capture the intent behind it. However, there are many other cases where a policy does not capture its intent.

When examining firewall policies that we know can be neither inconsistent nor incomplete, several studies have revealed problems that indicate that a policy may not be the way it was intended [ASH04, Woo04, Woo10]. Real-life firewall policies contain tens of thousands of clauses and are constantly updated to handle new threats/problems. This updating process is most often manual and prone to error. In addition, there are few to no mechanisms that will detect such problems [YMS+06].

From a logical point of view, we can view all those errors as weakening the condition of allowing/blocking a policy and deriving a contradiction. Most commonly, what is weakened is the ordering. Here we outline some logical formulation of the potential mismatches between policies and intent as previously identified. In all the cases we consider a single policy $\mathcal{P} : \langle p_1, a_1\rangle; \ldots; \langle p_n, a_n\rangle$. There are four types of anomalies that have been studied [ALRP12], which are summarized in Fig. 1. Each refers to two clauses, i and j, where we assume $1 \leq i < j \leq n$.

Type	actions	condition
Shadowing	$a_i \neq a_j$	$p_j \rightarrow p_i$
Generalization	$a_i \neq a_j$	$p_i \rightarrow p_j$
Redundancy	$a_i = a_j$	$p_j \rightarrow p_i$
Correlation	$a_i \neq a_j$	For some packet x, $x \models p_i \wedge p_j$

Fig. 1. Anomalies in Firewall Policies

Both shadowing and redundancy can make clause j vacuous – $\mathcal{P}$ is equivalent to one where a_j is replaced by the opposite action!. Thus, removal of the j^{th} clause would result in a policy whose allowed packets are exactly those of $A(\mathcal{P})$. It is, however, often the case that at least one of the clauses was inserted in the wrong position, and that the *intended policy* is not what is expressed by $\mathcal{P}$.

Fig. 2, borrowed from [YMS+06], describes examples of shadowing and redundancy anomalies. Each firewall policy is listed in the form ⟨*protocol srcIP srcPort dstIP dstPort*, *action*⟩, where *protocol* is either TCP or UDP, *srcIP* and *srcPort* are the source IP address and source port number, *dstIP* and *dstPort* are the destination IP address and destination port number, and *action* is either `block` or `allow`. Note that the source port and destination port are optional and not included in the examples discussed below.

In Script 1 of Fig. 2, rule 4 is shadowed by rule 2 because all of the packets that would be blocked by rule 4 have already been allowed by rule 2. Collective shadowing can also occur, such as rule 5 being shadowed by the combination of rules 1 and 3. Rule 5 allows *TCP 10.1.1.0/24 any*, but together, rules 1 and 3 block all of the packets that rule 5 would allow. In [YMS+06], shadowed rules are assumed to be anomalies because the inclusion of a specific allow (or block) rule is interpreted as an implication that the administrator intended a particular set of packets to be allowed (or blocked). In this example, rule 4 is assumed to capture the intent of the administrator due to its greater specificity compared to rule 2, and so the shadowing anomaly could be resolved by either removing rule 2 or by switching the precedence order of rules 4 and 2 in the list.

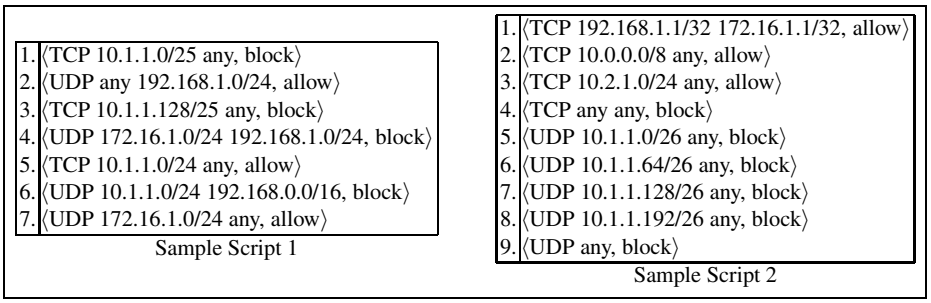

Fig. 2. Anomalies in Firewall Policies

Redundancy can occur between a pair of rules or among a group of rules. In Script 2 of Fig. 2, rules 2 and 3 are redundant because all of the TCP packets allowed by rule 3 have already been allowed by a preceding rule, rule 2. So rule 3 could be removed without affecting the behavior of the firewall. Redundancy also occurs between the group of rules (5, 6, 7, 8) and rule 9. If rules 5, 6, 7 and 8 are removed from the firewall script, then the UDP packets they block would still be blocked by rule 9 at the end of the list. The given firewall script could be simplified by removing rules 5 through 8, and the firewall would not change its action on any packets.

Generalization implies that p_j can be replaced by $p_j \wedge \neg p_i$ without impacting the semantics of the policy. It may also indicate that there is an error, where clause j should appear before clause i. In Script 1 of Fig. 2, rule 7 is a generalization of rule 4: the UDP packets blocked by rule 4 are a subset of the packets that would have been allowed by rule 7. To eliminate this generalization, rule 7 could be placed before rule 4, or rule 4 could be removed from the policy entirely.

Similarly, correlation implies that the only packets that "reach" clause j are those satisfying $p_i \wedge \neg p_i$ and thus p_j can be replaced by the above conjunction with no impact on the policy. However, since we assume that $p_n = \mathrm{T}$, unless a policy is trivial, correlation may be a "catchall" and a simple way to specify $\mathcal{P}$. Yet, in the case of $j \neq n$, correlation may indicate a mistake and that $\mathcal{P}$ does not specify the intended policy.

In each of these cases, a SMT solver can detect the existence of an anomaly and present a counterexample (packet) that demonstrates the anomaly, thus guiding an administrator to fix $\mathcal{P}$ in case of need.

Suppose that $\mathcal{P}$ has no unintended anomalies. Cascading it with another policy, say $\mathcal{Q}$, may have anomalies. Such anomalies can still be detected by SMT solvers, yet their resolution may be much harder. Assume an enterprise network where packets may be routed through different paths and different firewalls. Not only does one have to check for anomalies on a prohibitively large number of paths (this, in fact, is rarely the case in practice [YMS$^+$06]), but he or she also has to check the effects of "correcting" any of the firewalls in one path with the effects these changes will have on the other paths. Under the (unfortunately realistic) assumption that the number of clauses in each policy is in the tens of thousands, doing so for a real-life network requires much research and is beyond the scope of this paper.

In policies that are less precise than those of firewalls (HIPAA, for example) it is virtually impossible to even define anomalies. One may, however, be able to automatically detect cases of incompleteness or inconsistency. Perhaps it may be possible to guide a policy maker to avoid these things (note that if policies are first-order or beyond, no checker exists for detecting incompleteness and inconsistency). Yet, when faced with the need to intersect policies so as to be complete, contradictions are possible. Such contradictions may indicate human mistakes or they may suggest that a more precise description of intent is needed. In the following sections we describe a novel way for potential including intent in these considerations.

3 Policy and Intent

Even if all of the anomalies in a policy could be detected, there may still be discrepancies between the policy and the intent of the humans that created it. Ideally, a policy should perfectly reflect the intentions of the policy maker. Unfortunately, such a translation is not straightforward because such intentions exist in the mind of the policy maker. As a result, policy constructs may not be expressive enough to capture the policy maker's intentions, and this can lead to problems.

Consider a situation where a policy is designed such that a particular rule encompasses all of another potential rule. In such a situation, it will not be clear from the policy itself which rule the policy was intended to enforce. For example, with a firewall, a policy maker may add a rule to block all traffic on ports 6891–6900 to prevent file sharing with BitTorrent. However, these ports are also associated with file transfers in Windows Live Messenger. In such a situation, anybody reviewing the policy may not be able to infer which type of traffic the policy was intending to block because the policy notation is not expressive enough to capture that information.

Additionally, humans are not skilled at reasoning about complex systems such as policies and, thus, their intentions for how the system should work may themselves be inconsistent or incomplete [Nor83]. However, if the policy is not expressive enough to reflect the policy maker's intentions, these anomalies may not be detectable.

To illustrate this, let us once again consider the firewall example. When implementing or changing a policy, a policy maker may encounter a situation where he or she wants to implement multiple different changes, but one change contradicts all or part of others. For the firewall, this could occur if the policy maker needs to make two changes: one blocking all BitTorrent traffic and one allowing Windows Live Messenger. Whether or not the inconsistency in the policy maker's intent will manifest in the policy will ultimately depend on how the policy is implemented. If the rule for blocking BitTorrent's ports comes before the rule allowing Windows Live Messenger, the policy should be identifiable as inconsistent. However, if the rule allowing Windows Live Messenger comes before the rule blocking BitTorrent, then the policy will not be. In this last case, because BitTorrent and Windows Live Messenger can use similar ports, the policy can potentially allow BitTorrent traffic to pass through, in violation of the policy maker's intentions.

In many respects, these problems are inherent to the *expressive power* of the policy notation. Firewall policies can only specify policy rules based on IP addresses, ports,

and protocols, while it is often the intent to prevent or allow traffic based on applications or other contextual criteria. This is beyond the ability of firewalls because they don't allow these or other elements of a policy maker's intent to be considered. This is a common problem in design, analysis, and specification of systems, where it is imperative that the language or notation being used for representing system concepts be capable of representing the qualities of interest [Lev00].

Thus, to avoid these pitfalls, analysts and policy makers need a means of expressing and reasoning about policies that more closely aligns with the way humans form intentions. The cognitive science theory of goal directed behavior and planning asserts that actions are understood when they can be identified as being part of a task or plan, and, thus, a task or plan can be used with other situational information to infer what goals the human intends to achieve [EHRLR80, RJM88]. To reason about and perform analyses on goal directed human behavior and intentions, the human factors engineering community has developed task analytic methods.

When human factors engineers analyze complex systems, they use task analytic methods to understand how humans physically or cognitively perform tasks to achieve goals with the system [SCS00, KA92]. Task analysis is largely a manual process. An analyst will examine system documentation, engage in system training procedures, interview experts and users, and observe people interacting with the system. The output of this process will be what is called a task analytic model or task model. These can take a number of different forms. However, the typical structure represents human behavior as an input-output model. There are input variables representing information external to the human, human actions representing output, and a task structure and local variables that represent internal state. The task structure is typically represented as a hierarchy, where goal directed activities decompose into other goal directed activities and (at the lowest level) atomic actions. Task analytic modeling languages like ConcurTaskTrees [PMM97], Operator Function Model (OFM) [MM86], Enhanced Operator Function Model (EOFM) [BSB11, BB10], User Action Notation [HSH90], or AMBOSS [GMP+08] can be represented using discrete graph structures. In these models, strategic knowledge (usually represented as Boolean expressions using the model variables) that describes when the activity can execute, when it can repeat, and what the human expects to be true when it finishes. Finally, modifiers between activities or actions control how they execute in relation to each other.

Task models are relevant to this discussion for two reasons: they can be reasoned about formally and they can be used for intent inferencing.

Because of their discrete nature, tasks models can be represented formally and included in formal analyses. However, the majority of these analyses have focused on evaluating system safety properties in light of the modeled human behavior or on generating task-related usability properties for use in the formal verification of human-system interfaces (see [BBS12] for a review).

Intent inferencing is a process where an observer (who can be another human being or some automated process) attempts to develop a reasonable explanation for observed human behavior based on the current state of the system, a history of observed human actions, and knowledge about the plans the human operator uses to achieve goals (such as a normative task behavior model) [RJM88]. For example, the task analytic-based

OFM (Operator Function Model) [MM86] has been paired with a blackboard software architecture [HR85] to construct an automated intent inferencing system [RJM88]. In this implementation, the blackboard keeps track of the system's state and the actions a human has performed to determine what goal directed activities (contextually within the larger structure of an OFM) the human may be attempting to achieve. Because an OFM's activities can have strategic knowledge, this means that the intent inferencing system will have a boolean expression representing the state the human expects the system to be in upon completing the activity's goal.

In the same way that a task model can be used to describe how a human intends to accomplish goals with a system, a similar structure should be able to express how a policy maker intends to accomplish policy goals. In the following section, we describe what such a formalism might look like and outline how it could be used to evaluate policies.

4 A Task Analysis-Inspired Approach to Representing Policy

Task models represent the behavior (plan) humans intend to use to accomplish goals within a system as a hierarchy of goal directed activities and actions. Policies are similar in that they are meant to represent a policy maker's intent for how to accomplish specific outcomes in a regulatory context.

The concepts of activities and actions are not particularly relevant to policy since policy makers are not performing observable behaviors. However, there are policy analogs for both of these concepts. In task models, actions represent atomic behaviors a human can perform. For a policy, the closest equivalent would be an atomic implementation detail. Hence forth we will refer to this as a *rule*. In a task model, the main purpose of an activity is to accomplish a goal via human behavior. In a policy, we can simply assume the policy maker is attempting to achieve a policy goal or, simply, a *goal*.

A task model activity can have strategic knowledge that specifies when an activity should be performed (often called a precondition), what must be true when it terminates (sometimes referred to as a completion condition or postconditions), and other conditions specific to the fact that behavior is being performed. A policy goal would require a condition similar to a precondition to assert when it was relevant. We refer to this condition as *context*. A postcondition would also be relevant to a policy goal to allow it to assert what conditions will be required for the policy goal to be achieved. For the remainder of this paper we will refer to such a condition as *outcome*.

Fig. 3 summarizes the "translation table" between task models and policies.

Tasks	**Policies**
action	rule
activity	goal
precondition	context
postcondition	outcome

Fig. 3. Translation Table between Task Terminology and Policy Terminology

Conditions in task models are usually represented as Boolean expressions. However, for policy, these will likely not have the expressive power necessary for describing context and outcome conditions. What type of logic would be most appropriate for these is currently an open question and the source of much debate and research (see, for example, [BDMN06, RSMS10]).

Next, to be consistent with the cognitive theory that claims that the intentions of specific acts are understood in the context of a plan [EHRLR80, RJM88], we need to specify how goals and rules can be used to define a plan. Again taking inspiration from task modeling, where behavior is defined by hierarchies of activities and actions, we can define policy as hierarchies of goals and rules. In such an organization, a policy would be composed of a collection of high level, and presumably independent, goals. Each goal would decompose into lower level goals that would represent subgoals (each with their own context and outcome conditions) necessary for accomplishing their parent's goal. These goals could decompose into other goals and, at the bottom of the hierarchy, rules. In this way, each rule is explicitly associated with a hierarchy of goals describing what the policy maker intends to accomplish and in what context it is relevant. Alternatively, each policy goal (intention) has an explicit description of how it will be achieved.

In task models, every decomposition is modified by an operator that specifies how different activities and actions should execute in relation to each other: how many should be performed, whether they need to be performed in a specific order, and if they should be performed sequentially or can be performed in parallel. For matters of policy, it is not obvious what types of relationships will need to be enforced between goals and rules. It is clear from the firewall example that order is important in the enforcement of policy; however, other relationships may also be important.

The details of this modeling approach are far from complete. However, if successful, such a representation should allow for policies to be evaluated with respect to intent in ways not previously possible. Because it still uses rules as the basis for policy enforcement, these rules should still be capable of being reasoned about in completeness and consistency evaluations. However, contextualizing these rules in a hierarchy of goals should afford additional analyses that account for the policy maker's intent. Static analyses could be used to determine if the rules contained in a goal's decomposition will always achieve the goal's outcome condition. If the analysis indicates that this isn't true, the implication would be that the rules in the policy do not achieve the desired intent. Alternatively, static analysis could be used to examine the goals themselves to understand if there are contradictions between the outcome conditions of supposedly independent goals. This could be used to expose contradictions in intent that may not necessarily manifest as contradictions between rules (as per the firewall example given in the previous section). These are just some of the examples of the types of analyses such as a policy modeling approach could facilitate. Additional analyses may also be possible.

Once complete, this representation could prove to be a very useful for describing and evaluating policies. As in the case of complex systems, following a hierarchical development process has immediate and obvious benefits to newly developed systems. Leaving aside the question of tools, such a design process will undoubtedly results in

policies that can be analyzed, maintained, updated, and and enforced in a far superior fashion to the current state-of-the-art. However, just like the case of software engineering, there are issues of applying such a design process to legacy policies. This would require some "reverse-engineering" to obtain intent from policies, that, as we know, are extremely complex. This is reminiscent of the exact same situation in complex systems. In [Lev00], Leveson refers to an attempt to capture the requirements of TCAS II (an aircraft collision avoidance system), quoting from a report stating "the intent was largely missing" and that it is extremely difficult to derive post-priori rationale for decision made in the design.

Yet, in some cases, we can envision that some "legacy" policies can be reverse-engineered. For example, in the case of firewall policies, one can consult a reverse-look-up table that allows one to derive some potential intent behind filtering policies. Going back to the BitTorrent example, it is fairly easy to derive that blocking UDP/TCP communication on ports 6891–6900 effectively blocks both BitTorrent and Windows Live Messenger traffic. Thus, it is possible to automatically construct a candidate "intent document" for such policies. Obviously, it is ultimately a human being who must determine whether the automatically derived intent is consistent with his or her actual intent.

5 Conclusion and Future Work

In this paper, we introduced the notion of intent as the driving force behind policies and discussed the advantages direct reasoning on intent can have in the development and maintenance of policies that will be consistent, complete, enforceable, and maintainable. However, we have not proposed specific systems or mechanism through which to obtain our ultimate goal as this will be the topic of much further research. Rather, we have attempted to argue that such mechanisms are necessary and should be studied and developed.

Bringing reasoning about intent into the study of policies is only half of our vision, and the only one discussed here. The other part is the study of "consequences." Having contradictory or incomplete policies is unavoidable. The former is often a result of the need to intersect policies, as will be necessary when several entities, each with its own policy, have to derive a joint one. For example, consider a health policy that requires sanitization, or even destruction, of health records, and a malpractice insurance policy that requires saving such records. A physician must satisfy both, and will be left in a quandary because of the apparent contradiction. An "intent calculus" may assist in detecting such contradictions, yet, there may be no way to overcome them, and the entity that must make a decision (in our example, the physician) will need to decide which of the policies she is to violate. Such a decision requires reasoning about expected utilities. Unlike classical game theory, here utilities will not always easy to define and or will be impossible to order. Moreover, they will be based on subjective evaluations of different options. One physician may have different ethical values than another and may prefer to violate one policy over another. Even incompleteness may raise the same issues. For example, a health directive that refers to "medical professionals" and does not list who they are (physician assistants? medical secretaries? lab technician?) gives

rise to similar issues. Thus, reasoning about consequences may be a crucial part of decision making when faced with contradictory, or partial, policy decrees.

As mentioned several times in this paper, it is not clear what the best language is for expressing a policy given the policy's intent. Once again, as in the case of complex systems, a wrong choice of language may result in a policy that fails to capture its intent. This is obvious with the firewall policies, where the conditions expressible do not suffice to attain intended protection (as in the case of ssh or BitTorrents) – they may block (or allow) more than intended, and a sophisticated user may easily bypass them. Yet, using stronger languages (such as deep packet inspection) raises issues of ethics and norms, which are problems far and beyond those studied in complex systems.

In practice, one will need to have tools to support reasoning about policies and intents, both for the new and the legacy cases. Borrowing from software engineering, it may be possible to develop tools that will accomplish some of our goals using behavioral programming (for a review see [HMW12]), and variants of the Play Engine [HM03]. With these, one can describe "good" scenarios and, with the assistance of an automated system, refine the good/bad scenarios to the point of executable specifications, or, in our case, a policy with a formal intent model.

Such tools can assist in the much needed reverse-engineering of legacy policies. In Section 4 we outlined such a possible mechanism for firewall policies. Even with such policies, it may be difficult to obtain a succinct and readable intent specification from the policy itself, and the task may be much harder for more complex policies that are expressed in hundreds of pages of natural languages. Here, learning and natural language processing tools may be of assistance. However, we do not envision a fully automated system that can accomplish this goal. This is not unlike the case of complex systems where many of the methodologies used are essentially manual but can be assisted by automatic tools to accomplish certain subgoals.

Acknowledgement. We would like to thank Karl Levitt, Jeff Rowe, Andy Applebaum, Matt Bishop, and Modhi AlSobeihy for many fruitful discussions on the topics of policies. Special thanks are due to Tony Solomonides for introducing us to the European Directive for the healthgrid and some of its potential flaws.

References

[ALRP12] Applebaum, A., Levitt, K.N., Rowe, J., Parsons, S.: Arguing about firewall policy. In: Verheij, B., Szeider, S., Woltran, S. (eds.) Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12. Frontiers in Artificial Intelligence and Applications, vol. 245, pp. 91–102. IOS Press (2012)

[ASH03] Al-Shaer, E., Hamed, H.: Firewall policy advisor for anomaly detection and rule editing. In: Proc. IEEE/IFIP 8th Int. Symp. Integrated Network Management, IM 2003, pp. 17–30 (March 2003)

[ASH04] Al-Shaer, E., Hamed, H.: Discovery of policy anomalies in distributed firewalls. In: INFOCOM (2004)

[BB10] Bolton, M.L., Bass, E.J.: Formally verifying human-automation interaction as part of a system model: Limitations and tradeoffs. Innovations in Systems and Software Engineering: A NASA Journal 6(3), 219–231 (2010)

[BBS12] Bolton, M.L., Bass, E.J., Siminiceanu, R.I.: Using formal verification to evaluate human-automation interaction in safety critical systems, a review. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans (in press, expected 2012)

[BDMN06] Barth, A., Datta, A., Mitchell, J.C., Nissenbaum, H.: Privacy and contextual integrity: Framework and applications. In: Proceedings of 27th IEEE Symposium on Security and Privacy (May 2006)

[BL75] Bell, D., LaPadula, L.: Secure computer system unified exposition and multics interpretation. Technical Report MTR-2997, MITRE Corp., Bedford, MA (July 1975)

[BSB11] Bolton, M.L., Siminiceanu, R.I., Bass, E.J.: A systematic approach to model checking human-automation interaction using task-analytic models. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 41(5), 961–976 (2011)

[EHRLR80] Erman, L.D., Hayes-Roth, F., Lesser, V.R., Reddy, D.R.: The hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. ACM Computing Surveys 12(2), 213–253 (1980)

[GMP+08] Giese, M., Mistrzyk, T., Pfau, A., Szwillus, G., von Detten, M.: AMBOSS: A Task Modeling Approach for Safety-Critical Systems. In: Forbrig, P., Paternò, F. (eds.) HCSE/TAMODIA 2008. LNCS, vol. 5247, pp. 98–109. Springer, Heidelberg (2008)

[Gut00] Guttman, J.D.: Security Goals: Packet Trajectories and Strand Spaces. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 197–261. Springer, Heidelberg (2001)

[HM03] Harel, D., Marelly, R.: Come, let's play: Scenario-based programming using LSCs and the play-engine. Springer (2003)

[HMW12] Harel, D., Marron, A., Weiss, G.: Behavioral programming. Commun. ACM 55(7), 90–100 (2012)

[HR85] Hayes-Roth, B.: A blackboard architecture for control. Artificial Intelligence 26(3), 251–321 (1985)

[HRU76] Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Communications of the ACM 19(8), 461–471 (1976)

[HSH90] Hartson, H.R., Siochi, A.C., Hix, D.: The UAN: A user-oriented representation for direct manipulation interface designs. ACM Transactions on Information Systems 8(3), 181–203 (1990)

[KA92] Kirwan, B., Ainsworth, L.K.: A Guide to Task Analysis. Taylor and Francis, London (1992)

[Lev00] Leveson, N.G.: Intent specifications: An approach to building human-centered specifications. IEEE Transactions on Software Engineering 26(1), 15–35 (2000)

[MM86] Mitchell, C.M., Miller, R.A.: A discrete control model of operator function: A methodology for information display design. IEEE Transactions on Systems Man Cybernetics Part A: Systems and Humans 16(3), 343–357 (1986)

[Nor83] Norman, D.: Some observations on mental models. In: Gentner, D., Stevens, A.L. (eds.) Mental Models, pp. 7–14. Lawrence Erlbaum Associates, Mahwah (1983)

[PMM97] Paternò, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A diagrammatic notation for specifying task models. In: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction, pp. 362–369. Chapman and Hall, Ltd., London (1997)

[RJM88] Rubin, K.S., Jones, P.M., Mitchell, C.M.: OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. IEEE Transactions on Systems, Man and Cybernetics 18(4), 618–637 (1988)

[RSMS10] Rahmouni, H.B., Solomonides, T., Mont, M.C., Shiu, S.: Privacy compliance and enforcement on european healthgrids: an appraoch through ontology. Philosophical Transactions of the Royal Society (368), 4057–4072 (2010)

[SCS00] Schraagen, J.M., Chipman, S.F., Shalin, V.L.: Cognitive Task Analysis. Lawrence Erlbaum Associates, Inc., Philadelphia (2000)

[Woo04] Wool, A.: A quantitative study of firewall configuration errors. Computer 37(6), 62–67 (2004)

[Woo10] Wool, A.: Trends in firewall configuration errors: Measuring the holes in swiss cheese. IEEE Internet Computing 14(4), 58–65 (2010)

[YMS+06] Yuan, L., Mai, J., Su, Z., Chen, H., Chuah, C., Mohapatra, P.: FIREMAN: A toolkit for FIREwall Modeling and ANalysis. In: IEEE Symposium on Security and Privacy, pp. 199–213. IEEE Computer Society (2006)