Content-centric Information Protection in Cloud Computing

**Christopher C. Lamb\*, Gregory L. Heileman\***

\* Department of Electrical and Computer Engineering, University of New Mexico

|  |  |  |
| --- | --- | --- |
| **Article Info** |  | **ABSTRACT** |
| ***Article history:***  Received October 10th, 2012  Revised October 20th, 2012  Accepted October 30th, 2012 |  | Information security was becoming even more important as organizations migrated systems to third-party infrastructure providers. Once migrated, previously transparent network topologies, information paths, and systems infrastructure became more opaque. This loss of control when coupled with storage of corporate and personally sensitive information lead to significant increases in potential vulnerability. Herein, we discuss the current state of the art with respect to network information security, discuss the shortcomings of current designs, and propose a novel taxonomy of network-enabled usage control architectures that can solve sensitive information transmission problems. We then close with a description of our information-centric overlay network and discussion of our experience using this system to manage real-time sensitive information flow over commercial cloud systems. |
| ***Keyword:***  Security  Cloud Computing  Usage Management  Information-centric Networks |
| *Copyright © 2012 Institute of Advanced Engineering and Science.  All rights reserved.* |
| ***Corresponding Author:***  Christopher C. Lamb,  Department of Electrical and Computer Engineering,  University of New Mexico,  MSC01 1100, 1 University of New Mexico, ECE Bldg., rm 125, Albuquerque, NM 87131-0001.  Email: cclamb@ece.unm.edu | | |

1. **INTRODUCTION**

Current enterprise computing systems are facing a troubling future. As things stand today, they are too expensive, unreliable, and information dissemination procedures are just too slow. Current approaches to partitioning information are unable to migrate to cloud environments. Additionally, the current approach of controlling information by controlling the underlying physical network is too cost ineffective to continue. This leaves large government and commercial organizations concerned with avoiding the exposure of sensitive data in a very uncomfortable position, where they cannot continue doing what they have done, and cannot migrate to what everyone else is doing (1).

Generally, such systems still do not use current commercial resources as well as they could and use costly data partitioning schemes. Most of these kinds of systems use some combination of systems managed in house by the enterprise itself rather than exploiting lower cost cloud-enabled services. Furthermore, many of these systems have large maintenance loads imposed on them as a result of internal infrastructural requirements like data and database management or systems administration. In many cases networks containing sensitive data are separated from other internal networks to enhance data security at the expense of productivity, leading to decreased working efficiencies and increased costs (2).

Finally, the length of time between when a sensitive document or other type of data artifact is requested and when it can be delivered to a requester with acceptable need to view that artifact can be prohibitively long. These kinds of sensitive artifacts, usually maintained on partitioned networks or systems, can require extensive review by specially trained reviewers prior to release to data requesters. In cases where acquisition of this data is under hard time constraints like sudden market shifts or other unexpected conditional changes this long review time can result in consequences ranging from financial losses to loss of life.

Federal, military, and healthcare computer systems are prime examples of these kinds of problematic distributed systems, and demonstrate the difficulty inherent in implementing new technical solutions. They, like other similar systems, need to be re-imagined to take advantage of radical market shifts in computational provisioning. New approaches to networking and information management present possible solutions to these kinds of problems by providing distributed information-centric approaches to data management and transfer (3) (4). Cloud systems certainly exhibit economic incentives for use, providing cost savings and flexibility, but they also have distinct disadvantages as well (5).

How to address these issues is an open research question. Organizations ranging from cloud service providers to the military are exploring how to engineer solutions to these problems, and to more clearly understand the trade-offs required between selected system architectures (6). Within this paper, after reviewing the current state of the art in secure systems, we describe specifically how information can be better protected when transiting dynamic networks while still providing timely access to needed information. We present a specific taxonomy of development that demonstrates how to migrate from current to future systems, and describe our experience with our own information-centric overlay prototype. The specific contributions of this work include our taxonomy, our approach to applying information-centric security in dynamic networks, and the application of our ideas to not only current cloud-based systems but to information and content-centric networks as well (7) (8) (9) (10).

**1.1. Current Solutions**

The Unified Cross Domain Management Office (UCDMO) supports efforts to develop other specific solutions that have been presented over the past few years to handle this kind of information management. The National Security Agency set the standard in this area initially. In 2009, at a conference sponsored by the UCDMO, Booz Allen Hamilton (BAH) and Raytheon presented alternative notional architectures contrasting with current NSA-influenced approaches (11) (12) (13) (14). These cross-domain solutions are intended to enable sensitive information to easily flow both from a higher sensitivity domain to a lower sensitivity domain, and from lower to higher as well. They generally act over both primary data (say, a document) and metadata over that primary data as well. Note that in these systems, in most cases, human intervention is still required to adequately review data prior to passing into lower security domains.

The NSA conducted initial work in this area. Their standard-setting efforts culminated in reasonable conceptual system architecture, using groups of filters dedicated to specific delineated tasks to process sensitive information (12).

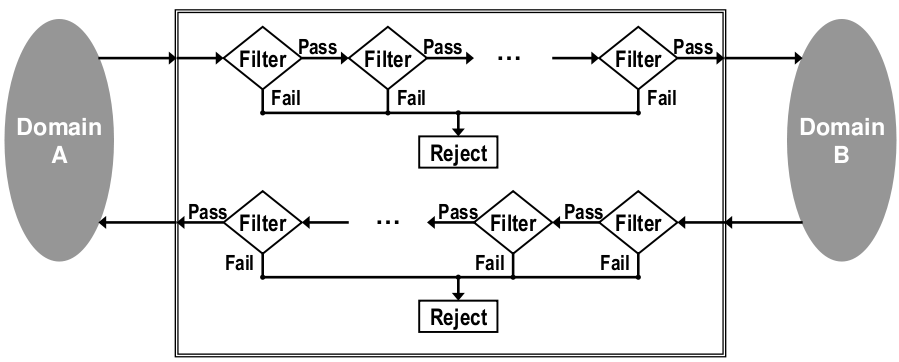


Figure 1. Original Cross-Domain Notional Architecture

In the scenario portrayed in Figure 1, Domain A could very well be a private cloud managed by the U.S. Air Force, while Domain B is a public operational network of some kind shared by coalition partners in a joint operation. A system user attempts to send a data package consisting of a primary document and associated metadata from Domain A to Domain B. At some point, that submission reaches a guard, which contains at least one filter chain. Each filter chain then contains at least one filter. Individual filters can execute arbitrary actions over a submitted data package and have access to any number of external resources as required. At any point, a filter can examine the data package and reject it, at which point it will frequently wait for human review. If a filter does not reject a data package, it passes that package onto the next filter or submits it for delivery to Domain B.

In recent years, the NSA has extended the legacy system architecture for cross-domain information sharing to exploit service-oriented computing styles (12). Visualized in Figure 2, this model incorporates more modern conceptual elements and componentry.

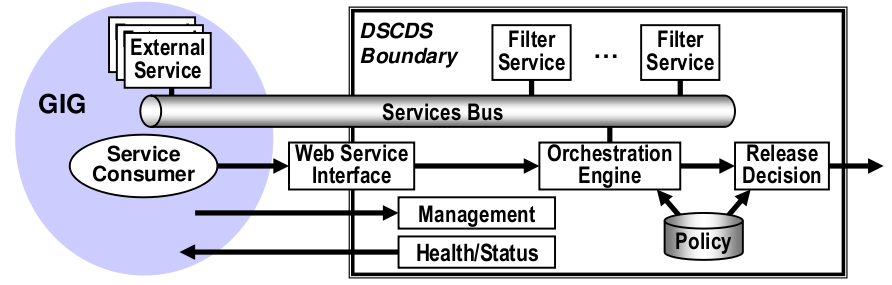


Figure 2. Service-centric Cross-Domain Notional Architecture

We see on the left the Global Information Grid, or GIG. On the right, we have the Distributed Service-oriented Cross Domain Solution, or DSCDS. The GIG is not a truly open system --- rather, it is a loosely coupled collection of computational services handing data at a variety of levels of sensitivity, federated to provide stakeholders timely access to relevant information (11). The DSCDS is essentially the embodiment of the NSA's cross-domain vision applied to service oriented computing. This model fuses various technology choices with previous cross-domain thinking.

Indicative of this more modern system design, we have a variety of services and service consumers attached to a common service bus within the GIG. Within the DSCDS, we have groups of filters implemented as services inspecting transferred data when moved over the bus. Finally, all of this interaction is managed by a management interface and controlled by an orchestration engine accessing a centralized group of policies. Note that here we have begun to access a common policy repository for various types of security metadata regarding primary data elements.

In the past few years, Raytheon has offered a new model for cross domain use influenced by the NSA service-oriented model (14).

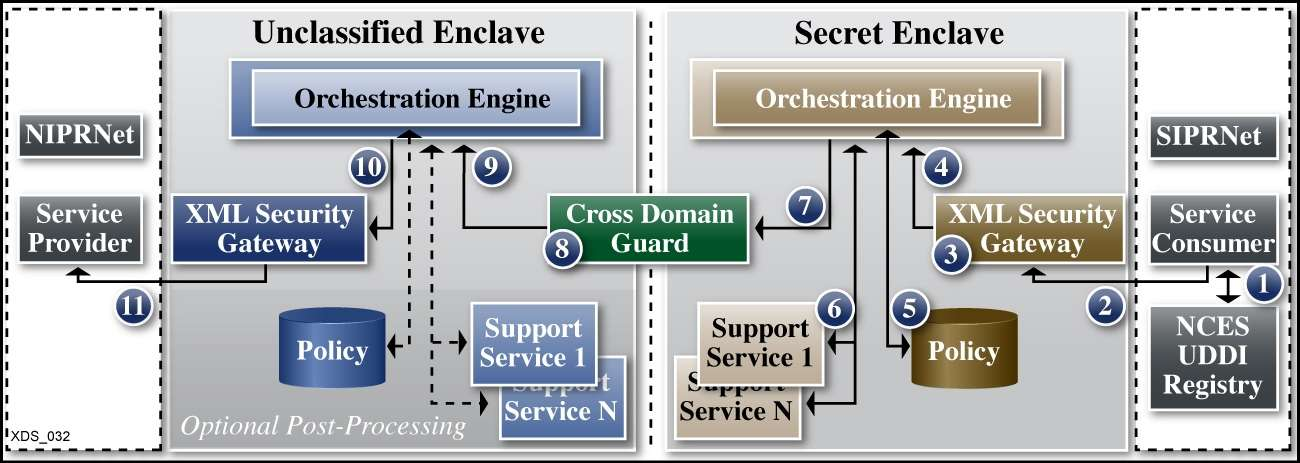


Figure 3. Raytheon’s Notional Architecture

The model in Figure 3 is more grounded in the actual technical environment this kind of solution would be embedded within. Here, we have the Non-secure Internet Protocol Router Network (NIPRNet) as one domain, and the Secret Internet Protocol Router Network (SIPRNet) as the other. Here, NIPRNet is the lower security domain (lowside), and SIPRNet the higher security domain (highside). This particular view shows the motion of data from the high side (SIPRNet) to the low side (NIPRNet). Here, a data request is submitted from SIPRNet first to the XML Security Gateway which calls into the Orchestration Engine for policy validation. The Orchestration Engine then coordinates calls into a Policy Repository as well as to a collection of external Support Services. Once rectified against these elements, the request is passed into the Cross Domain Guard that routes the request into the Unclassified Enclave in NIPRNet. Here, the request is passed directly through the lowside XML Security Gateway, without rectification, onto the Service Provider. The response from the Service Provider is then passed back to the requester via the inverse path. This model begins to use a centralized policy repository, just as the NSA Service Model. It also uses a single cross domain guard to transfer information from both the highside to the lowside, and vice-versa.

BAH submitted a competing model, also in 2009 (13). In fact, both Raytheon and BAH presented their models under competitive contract to the UCDMO at the same conference, so the domain application is not coincidental.

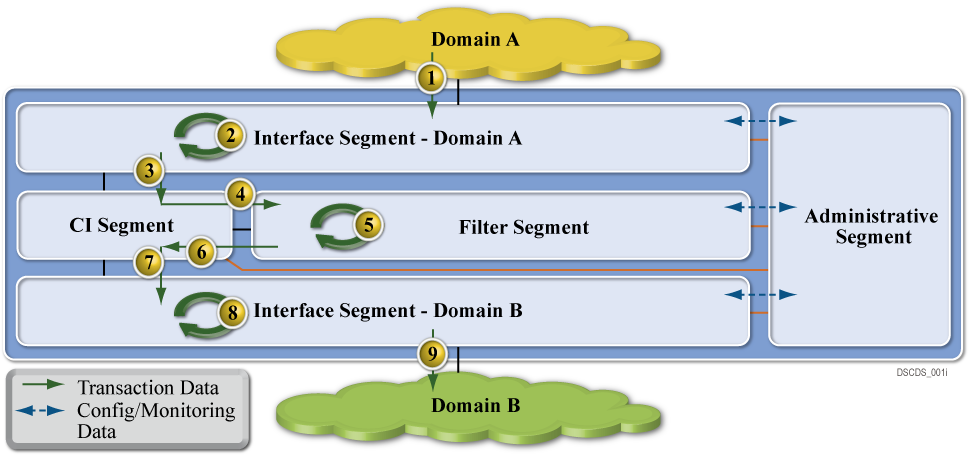


Figure 4. Booz | Allen | Hamilton’s Notional Architecture

Figure 4 embodies BAH's thinking with respect to cross domain information management. We have a Domain A as a high security domain, and Domain B as a low security domain. Here, we again have dataflow from the highside to the lowside through the cross domain management system. While not as detailed as the Raytheon proposal, this does have similar elements. Here, we data first travels from Domain A into the Interface Segment for Domain A, similar to the secret enclave used in the Raytheon model. From there, it moves into the CI Segment, which in turn submits the transferring data into the Filter Segment. From there, the package is moved into the Interface Segment for Domain B, and then onto Domain B. The Administrative Segment provides management and oversight of the system as a whole. Note the absence of specific policy-centric elements. This system is reliant on specific policy-agnostic content filters as well.

These kinds of cross-domain solutions still have clear similarities, and in fact have not progressed far beyond the initial notions of how these kinds of systems should work. They still, for example, all use some kind of filter chaining mechanism to evaluate whether a given data item can be moved from a classified to an unclassified network. Both NSA models used filters explicitly, as did the BAH model. They all use a single guard as well, a sole point of security and enforcement, providing perimeter data security, but nothing else. In each of these current system architectures, users are only allowed to exchange one type of information per domain. The physical instantiations of these models are locked by operational policy to a single classification level. Users cannot, for example, have Top Secret material on a network accredited for Secret material.

Future systems will generally demonstrate decentralized policy management capabilities, infrastructural reuse, the ability to integrate with cloud systems, and security in depth. Policy management will need to be decentralized and integrated within the fabric of the system. The system is both more secure and resilient as a result, better able to control information and operate under stressful conditions. Multi-tenancy can lower costs and increase reliability. An appropriately secured system facilitates integration of computing resources into multi-tenant environments. The ability to handle multi-tenant environments and to reliably secure both data at rest and data in motion leads to computational environments deployable in cloud systems. Finally, systems must operate under all conditions, including when they are under attack or compromise (15).

This work introduces the notion of usage management embedded in a delivery network itself. It also provides an in-depth analysis of the challenges and principles involved in the design of an open, inter-operable usage management framework that operates over this kind of environment. Besides referencing the material we have covered in depth to portray the current state of the art, the analysis includes application of well-known principles of system design and standards (16) (17) (18), research developments in the areas of usage control (19) (20), policy languages design principles (21), digital rights management (DRM) systems (22), and interoperability (23) (24) (25) (26) (27) towards the development of supporting frameworks.

While a large body of work exists on how overlay networks can use policies for network management, very little work has been done on using usage policies for content management. The primary contribution in this area focuses on dividing a given system into specific security domains which are governed by individual policies (28). This system fits into our proposed taxonomy as an α-type system as it has domains with single separating guards. A large body of work currently exists with respect to security in and over overlay networks as well. These kinds of techniques and this area of study is vital to the production development and delivery of overlay systems, but is outside the scope of this work.

1. **PROPOSED SOLUTIONS AND TAXONOMY**

A clear taxonomic organization of potential steps in approaching finer-grained policy based usage management helps in describing the difficulties inherent in developing potential solutions as well as aiding in planning system evolution over time. Here, we have four distinct types of integrated policy-centric usage management systems, as shown in Table 1. Of these four, only the first two levels are represented in current system models.

Table 1. Taxonomic Elements

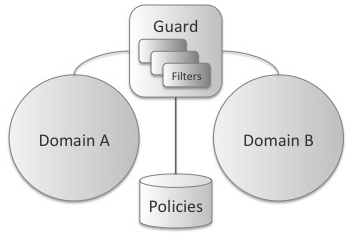
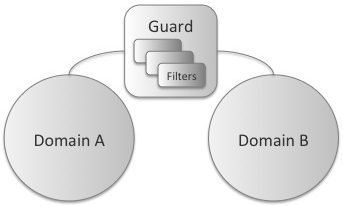
|  |  |
| --- | --- |
| Name | Description |
| ϕ | The initial level of this taxonomy, these systems have a single guard without policy-based control |
| α | These systems have begun to integrate policy-based control |
| β | Systems that have begun to integrate policy-based control with router elements |
| γ | Systems that have integrated policy-based control with routing and computational nodes |

In this taxonomy, it is not required that systems pass through lower levels to reach higher ones. This taxonomy represents a continuum of integration of usage management controls. Systems can very well be designed to fit into higher taxonomic categories without addressing lower categories. That said however, many of the supporting infrastructural services, like identification management or logging and tracing systems, are common between multiple levels. The taxonomy itself starts with the current state, integrating policy evaluation systems into the network fabric gradually, moving away from filters, then by adding policy evaluation into the routing fabric.

The ϕ classification consists of systems like the initial NSA and BAH notional models. These systems consist of two distinct domains, separated by a filter-centric single guard. The initial NSA system model is clearly of this type, separating two domains with a guard using filter chains. The BAH model is also of this type, using a Filter Segment to evaluate data packages transmitted between interface segments attached to specific domains.

Generally one of the domains supports more sensitive information than the other, but that is not always the case. In the models we have examined this has certainly been true, but classified information for example is commonly stored in compartments which are separated by clear need-to-know policies enforced by access lists and classification guides. These kinds of compartments contain information at similar levels of classification, but contain distinct informational elements that should not be combined.

In these kinds of systems, specific rules regarding information transfer and domain characterization are tightly bound to individual filter implementations. They are based on a priori knowledge of the domains the guard connects, and therefore are tightly coupled those domains. Furthermore, the filter elements are standalone within the system, in this classification, not availing themselves of external resources. Rather, they examining information transiting through the filter based purely on the content of that information. The set of filters that could be developed and deployed within the guard are unlimited. Developers could easily create a filter that inspects and possibly redacts the sections within the document, rather than passing or not passing the entire document through the guard. Indeed, if we assume even very limited processing capabilities within the guard, that is, Turing completeness, then this guard can be made as powerful as any solution we can derive for implementing a cross-domain solution (CDS). Thus the computational power of the guard is not the issue. The real issues are the benefits that can be gained by distributing the capabilities intelligently within the networked environment.



(a) ϕ classified structure (b) α classified structure

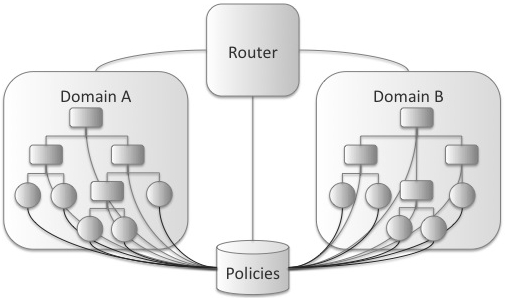
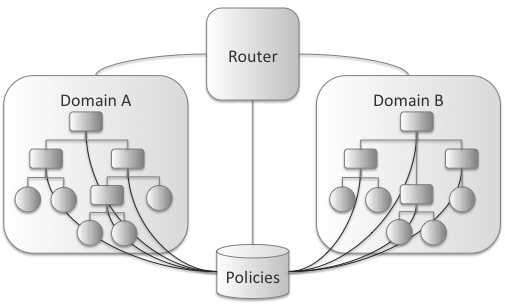
Figure 5. ϕ and α Conceptual Interaction

The α overlay classification contains systems that have begun to integrate policy-centric usage management. Both policies and contexts are dynamically delivered to the system. The dynamic delivery of context and policies allows these kinds of systems more flexibility with policy evaluation. The α category begins to integrate policy-centric management rather than using strict content filtering. Here, we again have at least two domains, Domain A and Domain B, though we could potentially have more. ϕ type systems require domain specific information to be tightly coupled to the filter implementations. Separating the permissions, obligations, and other constraints from the filters and incorporating them into a specific separate policy entity frees the guard from this coupling and provides additional flexibility to the system. The guard can continue to use filters to process data. These filters however are now more generic and decoupled from the specific domains the guard manages. The choice of using a specific filtering model rather than some other kind of construct is a design detail level to implementers. That said however, individual filters will be remarkably different and still need to understand the ontologies over which specific licenses are defined rather than specific content semantics. The policy repository is key to the implementation and differentiation of this taxonomy category. This repository can be implemented as a separate repository keyed into via a data artifact's unique URI, for example. It could also represent a policy sent in tandem with a data artifact in a data package. The policy repository may be implemented as some kind of external service, and as such, represents the first such external service explicitly used in this taxonomy. Other external services may well exist and be used to adjudicate information transfer decisions as well.

The β taxonomic category begins to integrate policy-centric processing with router elements in a given network. While this work is centered on using overlay technology to illustrate and implement these concepts, it is important to note that this kind of distributed policy-centric processing could very well be distributed into the physical routing fabric of a given network as well by extending Software Defined Networking systems like OpenFlow (29).

In this model we can also host multiple domains as a result of flexible policy-based content examination. Each domain hosts a network of some kind, though that hosted network could very well be a degenerate network of a single system. Each network hosted in a domain is hierarchical, with specific computational nodes embodied by workstations, tablet computers or mobile devices, and routing points embodied by routers or switches of some kind.

Policy evaluation in this model has begun to penetrate into the routing elements of the specific domain networks. We have started to penetrate into the routing fabric of the network by doing content evaluation at router points. Content-based switching networks have been successful in other domains, and such techniques can be used here to provide policy evaluation capabilities (30). Certain types of traffic are easier to evaluate than others however. For example, HTTP requests and responses are easier to examine that TCP packets. When examining TCP packets, systems generally require additional context to select an appropriate packet window (e.g. the number of packets cached for examination). HTTP traffic does not usually require this kind of complexity.



(a) β classified structure (b) γ classified structure

Figure 6. β (on the left) and γ (on the right) Conceptual Interaction

This migration of policy evaluation into the routing fabric provides for enhanced data security and better network management, especially if part of a network is compromised. Now that policy decisions can be made at the router level in a given network, we are starting to have network security in depth rather than simple perimeter protection. This not only provides the ability for additional information protection, but also allows for different compartments holding information at different need-to-know levels to be created ad-hoc under different routing segments. In cases of network compromise, this kind of dynamic policy enforcement can also allow for quick node excision as well.

The γ compartment has integrated policy evaluation with compute and routing nodes. Here, policies can be evaluated against content at all network levels --- nodes emitting requests, nodes fielding requests, and all routing elements in between. The policy repository is supplying services to all computational elements in both domains. This gives us increased granularity with respect to data compartmentalization by integrating information security into each network element. At this point, the network can create compartments of single nodes, while previously in β level systems compartments could only be created under specific routing elements. At this level, we can also provide services revoking data access based on policy evaluation decisions when needed. Furthermore, individual node exclusion is possible as well. β classified systems could excise network elements under specific routers by dynamic policy application. Now, we can apply the same functionality to individual compute nodes. For example, if a networked device like a smart phone is compromised, that device can be removed from access quickly or used to supply misinformation.

The various levels of the taxonomy vary primarily with respect to the inclusion of policy-based usage management and overlay structure. ϕ type systems are not structured with distributed use in mind, nor do they use policy-centric management. Conversely, γ type systems are both purely policy oriented and completely distributed. As systems move through the various levels of the taxonomy they gradually move from one side of the spectrum to another. Distributed usage management structures, hierarchical or otherwise, gradually migrate into the network beginning with β systems. Policy orientation is injected into the architectures starting with α systems and moving into the network fabric in parallel with overlay inclusion.

1. **RESULTS AND EXPERIENCE**

At this point, we have created and deployed baseline system images in both Amazon's Elastic Compute Cloud (EC2) and Rackspace Servers infrastructures. We have also created and exercised our deployment, configuration, and logging systems to enable distributed monitoring and centralized reporting. Overall, we currently have 20 nodes running with two distinct providers geographically dispersed across the continental United States. This leads to a distinct requirement for a centralized system with distributed access for initial configuration information as well as logging and auditing. We have implemented this infrastructure using Amazon's Simple Storage Service (S3), accessible from both Rackspace and Amazon hosted virtual machines.

The specific technical components are EC2, S3, Rackspace Servers, and GitHub. Both EC2 and Rackspace nodes are Ubuntu virtual machines, albeit at different versions, as we run Ubuntu version 11.04 in Rackspace and Ubuntu Version 12.04 in Amazon's infrastructures. These systems are provisioned with Git, Ruby, the Ruby Version Manager (RVM), and supporting libraries. They all run as micro-instances or equivalent, and are bootstrapped with the appropriate project information to begin to participate as an overlay network node. While EC2 and Rackspace Server infrastructures are infrastructure-as-a-cloud (IaaS) offerings supporting virtual machine instances of various types, Amazon S3 is a simple key-value store. Running with REST sematics over HTTP, S3 stores arbitrary documents associated with specific keys in buckets. Any authorized participant can download these documents, where authorization state is proven by possession of a secret key. In this way, we can store the global configuration of a specific overlay network in a single location from which every node can access informationm with respect to their pending role and needed configuraiton information. Likewise, all overlay network state can also be saved to centralized buckets for later analysis. Finally, Github is a centralized source code repository used to share code between all participating nodes. Prior to each content network instantiation, each node checks the repository for updates, and downloads them if they exist.

All data saved within S3 is serialized in a text-based data serialization language known as YAML. YAML is a widely supported hierarchical data representation language with support within the Ruby core platform. This enables us easily serialize Ruby-native data structures to text-based representations for storage within S3. More importantly, it simplifies post-experimental data analysis as any information logged to the centralized logging system during a given experimental run can be easily read and analyzed after the fact. We use Capistrano to manage and initialize overlay nodes. Capistrano is a distributed deployment system initially used to manage large clusters of Ruby-on-Rails systems. It has since expanded into a general-purpose distributed deployment toolchain, tightly integrated with Git. This allows us to bootstrap different configurations of networks from a single command-and-control node simply and efficiently.

The unique strength of this system is enabling dynamic distributed content control. This includes information retraction, redaction, protection, and secure routing. Information retraction involves quickly removing a user's access to sensitive data. Redaction addresses simple data removal, while protection would operationally involve applying encryption layers of increasing strength based on operational demands. Finally, secure routing would provide the ability to send data over a more secure link if such a link is available and required.



Figure 7. Envisioned Network Node Interaction

In this system information retraction involves changing the execution context such that access for a given user, perhaps even on a specific device, is removed. This context then propagates through the information network and attached clients. This is useful when a given user, say a coalition partner, is suddenly considered compromised and can no longer be allowed access to sensitive information. Likewise, a specific user's system may likewise be compromised and be forbidden access to specific information.

Information redaction is generally used when a user simply does not have authorization for a specific section of content, generally within a larger document. In these cases, that information and related policy metadata are simply removed from any query responses. Likewise, information protection also addresses specific subsections of information in a larger document, but unlike redaction, a user is in these cases authorized to access information, but one of the links over which the information must travel is not authorized to transmit specific sensitive information. In these cases that information can be encrypted with appropriately strong encryption to allow for more secure information transmission.

Finally, secure routing use directly addresses the ability to select communication links based on information content. In these situations, a network has more than one path over which to return content. Furthermore, these multiple paths have different characteristics providing different levels of service. The system, based on rules contained in a policy and the current context can then select communication links of different security levels when returning content.

The user interface subsystem shown in Figure 7 we use to drive typical use processes requests and returns information from both Google Maps and the content network based on those requests. Technically, it is based on the latest version of Ruby on Rails (RoR) using standard RoR configuration conventions running on top of Ruby 1.9.\*. We use Rake for deployment, and Gem for component installation. We use Bundler to maintain consistent application dependency state and RVM to manage Ruby virtual machine versions. HTML 5 interface elements are defined using Syntactically Awesome Stylesheets (SASS) and HTML Abstraction Markup Language (HAML).

The content network can be configured to run as an HTTP overlay system using HTTP routers and nodes or in a peer-to-peer configuration. In either case, queries can be submitted to the network from any one of the constituent nodes. In hierarchical networks, routers do not store data; rather, they focus solely on routing queries through a network. After initial submission, queries propagate throughout the network based on user-submitted search parameters.



Figure 8. Hierarchical Node and Router Structure

In both configurations, the common functional flow is built around responding to content queries with information of appropriate sensitivity for a given query context, as shown in Figure 8. In general, applications accessing the content network are designed with a layered perspective, with an application layer fielding initial requests, a protocol-agnostic domain layer that manages query responses, and an infrastructure layer that contains specific required libraries and other technical artifacts. In these systems, the application layer handles HTTP protocol issues, translating requests from the lingua franca of HTTP into the domain language reflected in the domain layer. The infrastructure layer consists of various data management technologies called upon by the domain layer when needed.

Figure 8 also highlights communication ordering within components in a hierarchical content network and also shows the functional components within the system. From a communication perspective, requests come in through the application layer and are then handed off for processing to the domain layer. The domain layer retrieves the current context and is responsible for query dispatch (in the case of a router) or data responses (in the case of a node) that are managed according to the current environmental context.

The primary components in the router and node systems' application layer are small adapters intended to translate between HTTP protocols and domain components. They are:

* **Context Manager Client Service (ctx\_mgr\_s)}** – This is an adapter between the domain context manager and the external context service.
* **Node Service (node\_s)** – The node service provides a RESTful interface to external clients. All content requests are initially sent to a known node service. This is essentially the external interface to a given content network. A content network generally contains many distinct nodes as well.
* **Router Service (router\_s)** – The router service is essentially a customized HTTP router that dispatches content requests and responses through a hierarchical content network in accordance with established policies and the current environmental context.
* **Dispatch Service (dispatch\_s)** – This service dispatches information requests to known nodes based on known policies and context.

The domain layer components include:

* **Context Manager (ctx\_mgr)** – The context manager client service calls into the context manager service to retrieve the most current contextual information with respect to the content network, attached clients, users, and devices.
* **Node (node)** – The node component contains all logic needed to process and respond to information requests. Nodes manage requests, responses, context evaluation, and usage management mechanism application.
* **Usage Management Mechanism (umm)** – The usage management mechanism will apply rules grouped into policies against a known context to determine the acceptability of an intended action. It will indicate whether or not that action can proceed. It can also make changes to a proposed action so that the alternative action can be executed.
* **Router (router)** – Router domain components manage the distribution of information requests and responses, applying managing information dispersal throughout a content network in accordance with context and policy.
* **Dispatcher (dispatch)** – Dispatchers send requests to known routers or nodes in the larger context network.

Finally, the sole infrastructure component is the **Information and Policy Repository (repo).** Unique to nodes, information and policy repositories contain specific network content, organized by key, and associated policies.

We use the same components to assemble non-hierarchical networks, in which nodes have both content and policy storage as well as request-response and dispatching responsibilities. Also note that context management and usage management components are shared between all types of content networks as well as all types of component systems within those networks. Non-hierarchical nodes and hierarchical routers and nodes all need these kinds of services.

This system uses attribute based mechanisms for usage management. The policies defined over content must therefore consist of rules that address usage over an ontology of possible user attributes of concern. We are specifically interested in a user's primary attributes: mission affiliation, clearance levels (both sensitivity and category), organization, and computational environment (consisting of both device and operating system). We also make decisions with respect to usage based on a secondary attribute, need-to-use.

Sets differ from orderings as sets denote membership with no associated value. Orderings on the other hand have distinct values increasing from left to right in the listed enumerations. For example, a user can be affiliated with a hypothetically specific mission in Domain A, either tropic\_thunder or gallant\_entry, or both. That user is also associated with a sensitivity value, either unclassified, secret, or top secret, where top secret is the most sensitive and unclassified the least, where top secret dominates all other classifications. Need-to-use decisions are based on the current context in tandem with mission and organizational affiliation. We use attribute based control in these scenarios, in which we make access decisions based on the attributes of a requesting user rather than defined roles. User attributes support defined policy elements. Not every policy attribute has a corresponding user attribute as not all policy attributes are associated with users. Some are associated with the user's environment, like operating system or device.

Policies are evaluated either via direct set membership or via membership in a category in an ordering. Content can be affiliated with multiple sets with regard to set-oriented attributes. Likewise, users can belong to multiple sets as well. Both content and users will be associated with a single value from an ordering element, as that value is inclusive of lower values as well. For example, a user can be affiliated with both the tropic\_thunder and gallant\_entry missions, but only one of the clearance values of uncleared, secret, or top secret. In the case of clearance values, secret subsumes uncleared, so a user with a secret attribute set would be able to access any unclassified material.

In the scope of this project, we use a Ruby-based domain specific language (DSL) to describe policies. In larger heterogeneous deployments, a standards-based alternative like XACML would be more suitable. This project however is not focused on developing a complete policy specification language, but rather on using one in a very dynamic environment. XACML, for example, is a very large and complete standard that would require a significant investment of effort to implement. It can also tend to be verbose. A simple DSL focused on our specific needs is a more efficient alternative that allows us to focus our time and effort on the goals of this work rather than implementation of a large standard.

|  |
| --- |
| **policy\_set {** |
| **policy(:p1) {** |
| **match :all** |
| **rule(:mission\_affiliation) { |x| x == :tropic\_thunder }** |
| **rule(:sensitivity) { |x| x == :top\_secret }** |
| **}** |
|  |
| **policy(:p2) {** |
| **include :p1** |
| **match :all** |
| **rule(:device) { |d| d == :workstation || d == :phone }** |
| **}** |
|  |
| **policy(:p3) {** |
| **include :p1** |
| **match :one** |
| **rule(:category) { |c| c == :vermillion }** |
| **rule(:organization} { |o| == :oceania }** |
| **}** |
| **}** |

Listing 1. Sample Policy Domain Language

Listing 1 is a sample of the DSL we use supporting a subset of XACML elements. In this example, we have a base policy, p1, which all other policies inherit. That policy requires that all rules evaluate to true. p2 adds another rule based on devices, all of which must evaluate to true as well. Finally, p3 adds two additional rules, only one of which must evaluate to true for the policy to be fulfilled.

**ACKNOWLEDGEMENTS**

Xx xxx

# REFERENCES

x

|  |  |
| --- | --- |
| 1. | Tallon PP. Understanding the dynamics of information management costs. Commun. ACM. 2010 May 1; 53(5): p. 121-125. |
| 2. | U.S. Department of Defense. Trusted Network Interpretation Environments Guideline USA: U.S. Department of Defense; 1990. |
| 3. | U.S. Department of Defense. Chief Information Officer, U.S. Department of Defense. [Online].; 2007 [cited 2012 September 29. Available from: <http://dodcio.defense.gov/Portals/0/Documents/InfoSharingStrategy.pdf>. |
| 4. | Hoover JN. Informationweek. [Online].; 2011 [cited 2012 September 29. Available from: <http://www.informationweek.com/news/government/cloud-saas/229401646>. |
| 5. | Pearson S, Benameur A. Privacy, Security and Trust Issues Arising from Cloud Computing. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on; 2010. p. 693 -702. |
| 6. | U.S. Air Force. SBIR/STTR Interactive Web Site. [Online].; 2011 [cited 2012 September 29. Available from: <http://www.dodsbir.net/sitis/archives_display_topic.asp?Bookmark=41198>. |
| 7. | Koponen T, Chawla M, Chun BG, Ermolinskiy A, Kim KH, Shenker S, et al. A data-oriented (and beyond) network architecture. SIGCOMM Comput. Commun. Rev. 2007 October 1; 37(4): p. 181-192. |
| 8. | Jacobson V, Smetters DK, Thornton JD, Plass MF, Briggs NH, Braynard RL. Networking Named Content. In Proceedings of the 5th international conference on Emerging networking experiments and technologies; 2009; New York, NY, USA: ACM. p. 1-12. |
| 9. | Ain Mea. D2.3 – Architecture Definition, Component Descriptions, and Requirements. Deliverable. Publish-Subscribe Internet Routing Paradigm; 209. |
| 10. | Ghodsi A, Koponen T, Rajahalme J, Sarolahti P, Shenker S. Naming in Content-oriented Architectures. In Proceedings of the ACM SIGCOMM workshop on Information-centric networking; 2011; New York, NY, USA: ACM. p. 1-6. |
| 11. | U.S. Department of Defense. Department of Defense Global Information Grid Architectural Vision. Informational. U.S. Department of Defense; 2007. |
| 12. | U.S. National Security Agency. Distributed Service Oriented Architecture (SOA)- Compatible Cross Domain Service (DSCDS) DSCDS Overview. In Unified Cross Domain Management Office Conference; 2009. |
| 13. | Booz | Allen | Hamilton. Distributed Service Oriented Architecture (SOA) Compatible Cross Domain Service (DSCDS). In Unified Cross Domain Management Office Conference; 2009. |
| 14. | Raytheon Corporation. Raytheon DSCDS Intro. In Unified Cross Domain Management Office Conference; 2009. |
| 15. | Ross R. Next Generation Risk Management. Unified Cross Domain Management Office. 2009. |
| 16. | Clark DD. The design philosophy of the DARPA Internet Protocols. SIGCOMM Comput. Commun. Rev. 1995 January 1; 25(1): p. 102-111. |
| 17. | Blumenthal MS, Clark DD. Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world. ACM Trans. Internet Technol. 2001 August 1; 1(1): p. 70-109. |
| 18. | Clark DD, Wroclawski J, Sollins KR, Braden R. Tussle in cyberspace: Defining Tomorrow's Internet. In SIGCOMM; 2002; Pittsburg, PA: ACM. p. 347-356. |
| 19. | Park J, Sandhu R. The UCON ABC Usage Control Model. ACM Trans. Inf. Syst. Secur. 2004; 7(1): p. 128-174. |
| 20. | Jamkhedkar PA, Heileman GL, Lamb CC. An Interoperable Usage Management Framework. In Proceedings of the Tenth ACM Workshop on Digital Rights Management; 2010; Chicago. |
| 21. | Jamkhedkar PA, Heileman GL, Martinez-Ortiz I. The Problem with Rights Expression Languages. In Proceedings of the Sixth ACM Workshop on Digital Rights Management; 2006; Alexandria, VA. p. 59-67. |
| 22. | Jamkhedkar PA, Heileman GL. The Role of Architecture in DRM Vendor Economics. In Satish D. Digital Rights Management: An Introduction.: ICFAI University Press; 2009. |
| 23. | Jamkhedkar PA, Heileman GL. DRM as a Layered System. In Proceedings of the Fourth ACM Workshop on Digital Rights Management; 2004; Washington, DC, USA: ACM. p. 11-21. |
| 24. | Heileman GL, Jamkhedkar PA. DRM Interoperability Analysis from the Perspective of a Layered Framework. In Proceedings of the Fifth ACM Workshop on Digital Rights Management; 2005; Alexandria, VA, USA: ACM. p. 17-26. |
| 25. | Koenen RH, Lacy J, MacKay M, Mitchell S. The Long March to Interoperable Digital Rights Management. Proceedings of the IEEE. 2004: p. 883-897. |
| 26. | Coral Consortium. Coral Consortium Whitepaper. [Online].: Coral Consortium; 2006. Available from: <http://www.coral-interop.org/main/news/Coral.whitepaper.pdf>. |
| 27. | Marlin. Marlin Architecture Overview. [Online].; 2006. Available from: <http://www.marlin-community.com>. |
| 28. | Perez GM, Clemente FJG, Skarmeta AFG. Building and Managing Policy-Based Secure Overlay Networks. In Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on; 2008. p. 597-603. |
| 29. | Openflow. Openflow - Enabling Innovation in Your Network. [Online].; 2011 [cited 2012 June 1. Available from: <http://www.openflow.org>. |
| 30. | JBoss. JBoss ESB. [Online].; 2011 [cited 2012 January 1. Available from: <http://www.jboss.org/jbossesb>. |

x

**BIBLIOGRAPHY OF AUTHORS**

|  |  |
| --- | --- |
| First author’s  Photo (3x4cm) | Xxxx (9 pt) |
|  |  |
|  | Faculty leader of ECE’s Informatics Laboratory, Greg Heileman also serves as an associate chair in the ECE Department as well as the director of the ECE undergraduate programs. In addition, he is the faculty advisor for the student branch of the IEEE. A senior member of the IEEE, his research interests are in information security, digital rights management, game theory and machine learning. During 1998 he held a research fellowship at the Universidad Carlos III de Madrid, and in 2005 he held a similar position at the Universidad Politécnica de Madrid. He is the author of the text Data Structures, Algorithms and Object-Oriented Programming, published by McGraw-Hill in 1996, and has more than 100 peer-reviewed publications. |