Identity Management and Access Control in Multi-clouds
Welcome,
Introduction and Administrative Issues

Matthew Scholl
Chief
Computer Security Division
NIST

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

8:30 am
Opening Keynote: Trusting Native Cloud Applications

Varun Talwar,
Co-founder and CEO
Tetrade
Tetrate NIST

Identity Management and Access Control in Multi-Clouds
Jan 23-24, 2020

Trusting Cloud Native Applications

Varun Talwar
@varungyan, vt@tetrate.io
Punching Bag @ Tetrate
Best practices for setting password

I changed all my passwords to "incorrect".

So whenever I forget, it will tell me "Your password is incorrect."
Today’s multi-cloud access/auth problem...

How to achieve Zero Trust Architecture in multi-cloud world?
Multi Cloud World
Modern Connected Enterprise

Access from anywhere, on any device and expect reliable and secure experience

Users

Employees

Distributed, Polyglot, Global, Using SaaS, Using OSS
Perimeter concept is officially dead

"The perimeter is no longer just the physical location of the enterprise [data center], and what lies inside the perimeter is no longer a blessed and safe place to host personal computing devices and enterprise applications [microservices]."

Keynote Session
Jan 23, 910 AM

“Zero Trust Architecture”
by Scott Rose

BeyondProd Document, from Google
https://cloud.google.com/security/beyondprod/
Multi Cloud

Service Centric World

DYNAMIC

DISTRIBUTED

FINE GRAINED
<table>
<thead>
<tr>
<th>Traditional Security</th>
<th>Cloud Native Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewalls, with internal communications considered trusted</td>
<td>Zero trust security with service-service communication verified and no implicit trust for services in the environment</td>
</tr>
<tr>
<td>Fixed IPs and hardware for certain applications</td>
<td>Greater resource utilization, reuse, and sharing, including of IPs and hardware.</td>
</tr>
<tr>
<td>IP based Identity</td>
<td>Service based Identity</td>
</tr>
<tr>
<td>Security specific requirements built into each application and enforced separately</td>
<td>Security built into common service stack</td>
</tr>
<tr>
<td>Services run in a known expected location</td>
<td>Services can run in anywhere in the environment</td>
</tr>
<tr>
<td>Limited oversight of security components</td>
<td>Centralized view of security policies and adherence to policies</td>
</tr>
</tbody>
</table>
Istio: The missing layer

Next gen infrastructure to connect, monitor, secure and manage services
SIDE CAR SERVICE
Service Sprawl

Credit: https://blog.twitter.com/engineering/en_us/a/2013/observability-at-twitter.html
Istio architecture

HTTP/1.1, HTTP/2, gRPC or TCP -- with or without mTLS

Config data to Envoys

telemetry

TLS certs to Envoys

Control plane API

Conference Session
Jan 23, 1200 PM
“Istio and Envoy” by Liam White
What can Istio security do for you?

- **Secure by default** - new and existing applications.
- Meet compliance obligations by **encrypting data in transit**.
- mTLS assures a secure, proven **service-based identity** for every call.
- With strong identity, **authorization** can be explicitly required.
NGAC: Next Generation Access Control

Fundamental relook at access control for modern interconnected enterprise
Role Based Access Control (RBAC)
- Easy to understand
- Easy to author correct policy
- Brittle, hard to model everything (role explosion!)

Attribute Based Access Control (ABAC)
- Hard to understand
- Hard to author correct policy
- (Infinitely) flexible, easy to model anything

Conference Session
Jan 23, 1045 AM
“NGAC for ZTA & Multi-Cloud”
by David Ferraiolo
Role Based Access Control (RBAC)
- Easy to understand
- Easy to author correct policy
- Brittle, hard to model everything (role explosion!)

Attribute Based Access Control (ABAC)
- Hard to understand
- Hard to author correct policy
- (Infinitely) flexible, easy to model anything

NGAC has the flexibility of ABAC, but has a set of guardrails that keep it structured and understandable.

Conference Session
Jan 23, 1045 AM

“NGAC for ZTA & Multi-Cloud” by David Ferraiolo
What can NGAC do for you?

- How to not have **role explosion**?
- **Who has access to what** resources across my enterprise, at any given time?
- Can I get a complete audit trail of all access control decisions?
- **Provable Security**: Why subject does/does not have access to certain resources
  - **Temporal policies**: Can permission be temporal and auto revoked?
  - **Location policies**: Can user/svc access data only in a region?
- **Enables Compliance** - PCI (e.g. card data), GDPR (e.g. EU location)
Combination leads us to a Zero Trust Architecture (ZTA)

Istio + NGAC

gives foundation for a true multi cloud, manageable zero trust platform, not bound to any cloud or network
Extending service mesh to data services

Conference Session
Jan 23, 230 PM

“Demo 1 & Demo 2” by Ignasi Barrera & Joshua Roberts
Unified Security Model

SP 800-204A(Draft)

Building Secure Microservices-based Applications Using Service-Mesh Architecture

https://csrc.nist.gov/publications/detail/sp/800-204a/draft
Thank you
Zero Trust Architecture

Scott Rose, Computer Scientist, NIST
Zero Trust 101
Scott Rose, NIST
scott.rose@nist.gov
A bit of background...

- Information Technology managers are facing an increasing number of cybersecurity breaches
- FCIOC recognized the need to develop understanding that enables the US government to adopt and transition to a Zero Trust Architecture
- Initiative and Steering Group was chartered in February 2019
- Partnering with NIST / NCCoE as the lead technical agency with involvement from a multi-agency project team
What is Zero Trust Architecture?

- First off, it’s really Zero *Implicit* Trust
- A way of planning a network and work flow
- Moving where policy decisions are made closer to resources.
  - Network location does not grant trust!
  - Access is granted per access, no blanket authentication
Tenets of Zero Trust

• All enterprise assets are considered resources.
• The enterprise ensures all owned assets are in their most secure state possible.
• All communication is done in a secure manner regardless of network location.
• Access to individual enterprise resources is granted on a per-connection basis.
• User authentication is dynamic and strictly enforced before access.
• All resource authentication and authorization is dynamic and strictly enforced before access is allowed.
• The enterprise collects as much information as possible about the current state of network infrastructure and communications and uses it to improve its security posture.
Two separate network planes:

- Control Plane: used by ZT components to set up and manage network
- Data Plane: used by applications for business processes
Identity Management and Governance in ZTA

• Users perform workflows
• Policy Engine needs to know who requester is and what attributes/privileges each account has before evaluating requests
• Access privileges should be as fine grain as possible
  • Accounts should only have privileges for resources that they need for workflow – no more
  • Separate organization “role” from access privileges in authorization
ZTA Relies on Many Current Federal Cybersecurity Initiatives

- NIST Cybersecurity & Risk Management Framework/FISMA – Planning
- FICAM – Identity Provisioning
- CDM – ID/Device/application management

- Smart Cloud and Update to Data Center Optimization Initiative (OMB M-19-19)
  - Cloud migration is main driver for ZTA

We’ve been moving to ZTA for years! (without knowing it)
Grey Areas – What’s Missing?

• Standardized interfaces between components
  • Risk of vendor lock-in

• What does a successful attack look like?
  • Not enough experience
  • Attackers will eventually figure out how to approach ZTA enterprises

• How will user activity change?
  • Asking for authentication more frequently may impact user behavior

• What about server-server communication?
  • Can’t do multifactor authentication between workloads/Automated Tech
Your Input is Welcome

- NIST SP 800-207 Zero Trust Architecture
  - Public comment period ends 11/22/2019

- Future NCCoE Demonstration Project
  - Winter 2019/Spring 2020

- Contact Information
  - Alper Kerman (NIST/NCCoE) – Zero Trust Technical Lead; Alper.Kerman@nist.gov
  - Scott Rose (NIST) – Zero-Trust Architecture Sub-team lead; Scott.Rose@nist.gov
  - Oliver Borchert (NIST) – Zero-Trust Technology Sub-team Lead; Oliver.Borchert@nist.gov
Questions & Feedback
The Yellow Brick Road to ZTA in the Cloud

Juanita Koilpillai, Founder and CEO
Waverley Labs
Software Defined Perimeter (SDP) is the most advanced zero trust implementation.

Juanita Koilpillai (CEO & Cloud Security Alliance SDP Working Group Lead)
AGENDA

Why Zero Trust (Value)

What is Zero Trust

What Zero Trust Addresses

Implementing Zero Trust

Introducing SDP - Zero Trust Orchestration
  - IaaS, SDN & IoT Examples, Agile ATO

Benefits of SDP
Why Zero Trust

Changing Perimeter - fixed network perimeter problematic for mobile devices

IP address conundrum – IP addresses simply provide connectivity; no user context; inherently open to compromises. Changes to IP addresses can mean extensive configuration, and errors creeping in to network security groups, and network access control lists.

Virtualized environments add new attack surfaces - but at scale.

Implementing integrated controls can be a challenge. Need a single pane of glass for systems to ‘know their security stance’.
Why Zero Trust – Threat Landscape

... Today, many paths exist to attack enterprises

- External Threats from all over the world...
- Insider threats within a user group (role).
- Insider threats, across user group boundaries.
Why Zero Trust – Threats Differ in the Cloud

.... Paths are hidden

- SaaS Trust Boundary
- PaaS Trust Boundary
- IaaS Trust Boundary

Application

Virtualization / Platform

Hardware

Facility/Physical

Why Zero Trust – Threats Differ in the Cloud

.... Paths are hidden

- SaaS Trust Boundary
- PaaS Trust Boundary
- IaaS Trust Boundary

Application

Virtualization / Platform

Hardware

Facility/Physical

Customers

System Owners / Providers
Why Zero Trust - State of Cyber Security

- Machine to Machine Connections FORCE securing machines
- Access to Services allowed BEFORE Authentication & Authorization
- Firewalls are Static - ONLY network information
What is Zero Trust

Allowing access to the network changes with “Zero Trust”; as the concept implies - users aren’t allowed any access to anything until they authenticate who they are.

A network architecture that withholds access until a user, device or even an individual packet has been thoroughly inspected and authorized.

Specifically, the least amount of necessary access is granted.

Continuous monitoring of suspicious user activity
What Zero Trust Addresses

- Access Control Vulnerabilities - access control mechanisms with current authentication and authorization protocols have weaknesses that are being exploited or bypassed.

- Endpoint monitoring Weaknesses - Vulnerabilities at the Network Layer prior to transport and application protocol and endpoint protection measures.

- Network Packet Inspection Limitations - Packet analysis happens at the application layer, so incursions can happen prior to detection.
Implementing Zero Trust

Requires authentication before access
- implicitly requires separate control and data planes
- immediate authentication

Requires ability to limit network connectivity and exposure
- drop network connections if authentication fails.

Requires granular trust mechanism
- unlike VPNs that do not have fine-grained access control
- implicitly requires authorization as well as authentication prior to access

Requires monitoring for suspicious activity
- implicitly requires instant knowledge of connectivity and use of services

Client provides device awareness
Firewall/Gateway provides network awareness
Application provides user awareness
Introducing SDP - Zero Trust Orchestration

1- Net facing servers/services hidden
2- Legit user given unique ID
3- Legit user sends the token (SPA)
4- Perimeter checks the token
5- Valid device + user = access
Zero Trust Orchestration

- Prevents attacks
- Integrates controls
- Provides visibility
- Focus on insider threat
SDP & IaaS – Bandwidth DoS Attacks

Test Results for DoS Attack

Fig. 5: Local SDP DoS resiliency test

Courtesy – Manhattan College
SDP & SDN – Flooding Attacks

Test Results for Flooding Attacks

Courtesy – Manhattan College

Fig. 2: Integrating SDP with SDN

Fig. 4: Network Traffic

Courtesy – Manhattan College
SDP & IoT – DoS Attacks

SDP Controller

Control Plane
Data Plane

AuthN + Encryption Key

SDP Gateway

IoT Devices

Publisher

SDP Client Device

Broker

MQTT

Sensor Data Input

Business Logic

SDP & IoT - DoS Attacks

Fig. 3. CPU Usage

Courtesy – Manhattan College
SDP & Agile ATO – Putting it all together

SDP Control Plane

HW Node

SW Node

User Control

Device Control

Service Control

SDN/SDP Gateway

Network/Storage

Scripts

Templates

SDP Data Plane

HW Node

SW Node

App

App

SDN/SDP Gateway

MW Node

ML

Algorithm

App

App

ML

ML

Algorithm

ML

ML

Algorithm

SDN/SDP Gateway

Network/Storage

HW Node

SW Node

User Control

Device Control

Service Control

SDN/SDP Gateway

Network/Storage

Scripts

Templates

LEGEND

- User environment and keys
- Device environment and keys
- Application environment and communication keys
- Network environment and firewall rules
- Storage environment and keys
Benefits of Zero Trust Orchestration

Reduced attack surface
- enhanced protection for cloud applications
- gives more centralized control to business/system owners to ensure Zero Trust architecture
- all authorized connections from whom, where, when, what monitored instantly because controls are integrated

Reduced cost of ownership
- reduce costs for endpoint prevention/detection
- reduce cost for incident response
- reduce complexity for integrating controls

Open Specification
- vetted by community
- hackathons
Coca Cola: Users access limited to a single connection to each authorized application - eliminating malware and information theft.

Coca Cola: Removing access to business applications on the internet is reducing attacks.

Google: Enabled BYOD and reduced the number of company laptops.

Avaya: Easier for SDNs to isolate authorized and unauthorized users/devices.

Verizon: Reducing attacks in Verizon cloud.
Continue the conversation . . .

Juanita Koilpillai
jkoilpillai@waverleylabs.com
linkedin.com/in/juanita-koilpillai-5551b111
http://www.waverleylabs.com/risk

Cybersecurity Assessments
SDP Design & Implementation
Definition of Failure Scenarios
Questions
(Slide 0)
Coffee Break
Back @ 10:45 AM
Next Generation Access Control for ZTA and Multi-cloud

David Ferraiolo,
Manager
Secure Systems and Applications
NIST
Next Generation Access Control (NGAC)* for ZTA and Multi-Cloud

David Ferraiolo
National Institute of Standards and Technology

*An ANSI/INCITS NGAC Standard.
RBAC challenges

Three primary issues.

• RBAC can be cumbersome to set up and manage
  • Difficulty of associating permissions with roles.

• Differentiating roles in different contexts (e.g., teller in branch A and
teller in branch B) can result in large quantities of role definitions.
  • In extreme cases, the number of defined roles can exceed the number of
    users.

• Role qualifiers alone are often insufficient for expressing real-world
  access control policies.
NGAC

• NIST has developed a specification and open source reference implementations of an authorization framework, referred to as the Policy Machine (PM).

• The PM is designed in support of, and in alignment with an emerging ANSI/INCITS standard under the title of Next Generation Access Control (NGAC).

• NGAC defines access control in terms of a fundamental and reusable set of relations and functions, following an attribute-based access control model.
Policy elements & relations

• Basic elements
  • Users, processes, access rights (resource and admin), operations (resource and admin), and resource objects

• Containers
  • User attributes, object attributes, and policy classes

• Relations
  • Assignments (define membership in containers, and containers of containers (form a Directed Acyclic Graph))
  • Associations (with assignments, used for deriving privileges)
  • Prohibitions (denies for users and processes access capabilities)
  • Event-pattern/admin-response (for dynamically altering the access state)
Assignments and associations

Users (u) are assigned to user attributes (ua), ua are assigned to ua, and ua are assigned to policy classes (pc). Objects (o) are assigned to object attributes (oa), oa are assigned to oa, and oa are assigned to policy classes (pc).

Users and objects are associated with an access right set (ars). Assignments and associations with respect to pc(s) are used to derive privileges (u, r, o), where r is an access right.
Administration

• Administrative capabilities via associations with admin access rights (aars)
• One admin can delegate aars to another though their admin rights to create associations
• Parameterized admin routines (E.g., Create File Management User (user, user name, user home)*)

(created relations, results of u2’s delegated capabilities)
NGAC Architecture

**Policy Enforcement Point**
**Resource Access Point**
**Policy Decision Point**
**Event Process Point**
**Policy Administration Point**
**Policy Information Point**

Note:
- Resource methods implemented in RAP
- Administrative methods implemented in PAP

When components can be implemented as microservices, a service mesh architecture can provide a robust assurance framework.
Policy Review and Resource Discovery

• NGAC supports policy review/analytics (e.g.,)
  • What are the objects a user can access?
  • Enables a Personal Object System (POS) for a user to display and select objects for which they are authorized access
  • Who can access an object?
  • Why can’t a user access an object?

• Access Review & Decisions through linear time algorithms
• Opens up the possibility for mapping into other access control structures (e.g., ACLs) and different types of resources
Multi-Cloud Deployment

- Cloud interprets RAP as a user with liberal permissions to NGAC created data
- Users centrally “see” cloud resources as logical entities via POS
- EPP centrally log access events

Personal Object System (POS):

- A users current access capabilities for objects in object attributes
Run-time stack for NGAC enabled applications

Methods for data presentation, data manipulation

Methods for read, write, and configuring access control data

DevSecOps can be used when NGAC Framework is implemented as microservices
Benefits and Features

• Reduced code base for data services
• Single sign-on
• Data interoperability among applications
• Comprehensive policy enforcement across applications
• Centralized audit

• Objects correspond to files maybe perceived as other data types, (e.g., messages, work items, fields, clip-board)
• Data never moves and only appears to be in application structures (e.g., folders, inboxes, records).
  • Share one file or data in bulk with similar efficiency
Reference Implementation

- **Data Services**: Office applications, file management, e-mail, workflow, records management, cut/copy-paste
- **Policies**: Tailored (through configuration) combinations of discretionary, mandatory, and history-based access controls:
  - DAC
  - RBAC
  - Time, location
  - Separation of Duty
  - Conflict of Interest
  - Workflow
  - Forms of confinement (read with restrictive write)
    - E.g., Only doctors can read medical records & MLS
    - Trojan resistant leakage
  - Read once, read one at a time
  - Non-repudiation
  - Tracking access - I know who can currently access to my data
NDAC

- Leverages NGAC and policy review for imposing access control over database queries as middleware
- Eliminates the need to implement and managing access control in applications and/or through the use of proprietary DBMS mechanisms
- Translates a user’s query to a permitted query for Select, and Grant/Deny for Update, Delete, and Insert
  - Users query may fetch entire data sets and NDAC restricts access to the set of data permissible for the user.
- Enforcement of policy combinations (e.g., DAC and RBAC), over DBMS data down to the field level
NGAC Authorization Engine

Policy Analytics

Access Control Data

NDAC components shown in red
Additional NDAC features

• Schema builder
  • For importing schema data and rows in the form of object attributes

• Policy Augmenter
  • For expressing policy in terms of those imported object attributes
Example Policy

• Staff can read Name and Phone fields of all records in Employee Table
• Staff can read all fields (SSN, Salary, Name, and Phone) and can write to Name and Phone fields in their own record
• Group managers can read all fields of all records in their group with the exception of the SSN field
Select for Bob (Grp1Mng)

Note: Bob can see the records he can access without issuing a query.
Implementation and Scale

• Policy configuration resides in PDP memory as a graph structure
  • Accommodates billions of nodes
• Updated from disk when policy changes
• Linear time algorithms for computing decisions and conducting policy review (over a subset of graph that pertains to the user)
Summary of Features

- Specify and enforce combinations of dynamic and static access control policies (e.g., DAC and RBAC) across virtual enterprise (VE)
- Policy analytics
  - E.g., who has access to what objects across VE
- Expression, enforcement, and delegation of administrative privileges over VE policy configuration
- Centralized audit of access events across VE
- Types of applications (Web Services)
  - Existing applications (e.g., office applications)
  - NGAC enabled applications (designed with NGAC in mind)
  - Database applications
Questions?

David Ferraiolo
david.ferraiolo@nist.gov
301-975-3046
SP 800-204A
(Public Draft) Guidelines for Deploying Service Mesh for Securing Multiservice-based Application Systems
Building Secure Microservices-based Applications Using Service-Mesh Architecture – NIST Special Publication 800-204A (Draft)

(Authors: Ramaswamy Chandramouli, NIST, Zack Butcher, Tetrate)

Ramaswamy Chandramouli (Mouli)

Zack Butcher
Building Secure Microservices-based Applications Using Service-Mesh Architecture

• Emerging Application Architecture in the Cloud - Microservices-based Application (MSA) Architecture

• MSA Security Requirements & Need for a dedicated Infrastructure

• Service Mesh – Architectural Layers & Functions

• Deployment Recommendations
Emerging Application Architecture in the Cloud - Microservices-based Applications Architecture

- Isolated fine grained services communicating via lightweight communication protocols
- Increasing adoption due to:
  - Platform Heterogeneity
  - Independent scalability
  - Deployment Agility
  - Availability of Tools
Microservices-based Applications – Security Requirements

• The sheer number of microservices results in more interconnections and more communication links to be protected.

• The ephemeral nature of microservices calls for secure service discovery mechanisms.

• The fine-grained nature of microservices calls for the ability to support fine-grained authorization policies.
Securing Microservices-based Applications –
Dedicated Infrastructure – Service Mesh

- Lessons learnt reveal that various security services for MSA must be tightly coordinated and policies consistently applied.
- This goal can only be achieved through a dedicated infrastructure.
- The Service Mesh has emerged as that dedicated infrastructure.
- Service Mesh has 2 architectural layers:
  - Data Plane, and a
  - Control Plane
Architectural layers of Service Mesh (Courtesy: O’Reilly)

Figure 1-1. Data plane versus control plane
Architectural layers of Service Mesh

- **Data Plane Functions:**
  (a) Handles cross-service communications
  (b) Enforces policies related to performance
      (Health checking, Timeouts, Retries,
       Circuit Breaking, Load Balancing)
  (c) Enforces policies regarding security
      (Encrypted Sessions, Authentication &
       Access Control)

- **Data Plane Realization:**
  Using a Service Proxy and in particular
  Side-car proxy (one for each service instance)
Architectural layers of Service Mesh (Contd..)

- **Control Plane Components & Functions:**
  (a) set of APIs, Tools for controlling and configuring Data Plane

  (b) Software for generating Certs, policies for authentication, authorization engine, gathering & aggregating telemetric data

  (c) APIs for providing resilience in network communication through features such as: load balancing, circuit breaking, rate limiting etc.
Deployment Recommendations for Service Mesh Components

The following areas are covered

• Communication Configuration for Service Proxies,
• Configuration for Ingress Proxies,
• Configuration for Access to External Services,
• Configuration for Identity and Access Management,
• Configuration for Monitoring Capabilities,
• Configuration for Network Resilience, and
• Configuration for Cross Origin Resource Sharing (CORS).
Communication Configuration for Service Proxies

(SM-DR1): Feature to specify protocols and ports into which a service proxy can accept traffic for its associated service

(SM-DR2): Features to limit access to services that a service proxy can reach – based on namespace and service name

(SM-DR3): Service proxy capabilities to support clients with different protocol than the target microservice

(SM-DR4): Ability to extend to Service proxy to implement use-case specific policies

(SM-DR5): Ability to swap to new dynamic policies at run time
Communication Configuration for Ingress Proxies & Access to External Services

**Ingress Proxies:** (SM-DR6): Feature for configuring traffic routing rules just like in service proxies – consistent policies from inside to edge.

**Access to External Services:** (SM-DR7): Access to external resources/service only by explicit policy. Modeled as service within service mesh and accessible by its discovery mechanism

(SM-DR8): Should have Network resilience (retries etc.) and access control features

(SM-DR9): Egress proxy features for configuring traffic routing rules just like ingress and service proxies as well as access and availability policies.
Configuration for Identity and Access Management

(SM-DR10): The identity of the microservice should be consistent (location agnostic) and unique

(SM-DR11): The signing certificate used by service mesh’s control plane should be rooted in the existing PKI’s root of trust and avoid the use of self signing cert for bootstrapping

(SM-DR12): The lifetime of a microservice’s identity certificate should be as short as possible – preferably on the order of hours

(SM-DR13): During identity certificate rotation, existing connections using the old certs should be gracefully retired

(SM-DR14): Certificates used to identify microservices should not be signing certificates
Configuration for Identity and Access Management (contd..)

(SM-DR15) : If the cert used for mTLS carries server identity, then a **secure name service that maps the server name to service identity** must be provided by a secure discovery service or DNS.

(SM-DR16): Each microservice should carry its own identity **rather than the identity of its namespace** (overriding the feature in some microservice run times that default to the identity of the namespace for a new service). This is critical since access policies must be applied at the level of a microservice rather than at the namespace level.
Configuration for Identity and Access Management (contd..)

(SM-DR17) : The policy scope for authentication should have the following options: (a) All microservices in all namespaces (b) All microservices in a particular namespace and (c) A specific microservice in a namespace

(SM-DR18): Identity tokens should be digitally signed and encrypted since claims included in them form part of the authenticated identity to build access control decisions
Configuration for Monitoring Capabilities
(Logging)

(SM-DR19) : Recommendation for Logging and Attack
detection capability – The proxy should log input validation
errors and unexpected parameter errors, crashes and core
dumps. Attack detection capabilities should include bearer token
reuse and injection attacks.

(SM-DR20): Recommendation for Logging Requests - The
proxy should at least log the common log format fields for
irregular request (i.e., non-200 response)

(SM-DR21): Recommendation for Log Message Content –
Log messages should contain at the minimum – Runtime/Stack
Information, Function Name, Line number at which the log
entry started and the Message
Configuration for Monitoring Capabilities (Metrics & Distributed Tracing)

(SM-DR22): Recommendation for Mandatory Metrics – The following should be the minimal set of metrics that should be gathered for external client and microservice calls –

(a) Number of client/service requests in a given duration,
(b) Number of failed client/service requests by failure code, and
(c) Average latency per service as well as average total latency per complete request lifecycle.

(SM-DR23): Recommendation for Implementing Distributed Tracing - Application services must be instrumented to forward the headers for communication packets they receive
Configuration for Network Resilience Techniques

(SM-DR24): Recommendation for Storing Data for Implementation of Network Resilience – Data pertaining to all configuration parameters such as – Retries, Timeouts, Circuit Breaker settings, Canary Deployments etc. should be stored in robust data stores such as Key/Value stores.

(SM-DR25): Recommendation for Implementation of Health Checking of Service Instances – the health checking function should be tightly integrated with service discovery function to maintain the integrity of the information used for load balancing (e.g., not including unhealthy hosts in load balancing pool)
Configuration for Cross Origin Resource Sharing (CORS)

(SM-DR26) : The CORS policy for the edge service (entry point for the microservice) must be configured using the service mesh capability (e.g., VirtualService resource’s CORS Policy configuration in Istio) rather than handling it through microservice application code.
Summary and Conclusions

- Service Mesh is an infrastructure layer for providing a set of comprehensive, consistent and coordinated set of support services.
- Most of the functionality is provided through sidecar service proxies without change to microservices code (for most part).
- Infrastructure functions include: (a) Secure Service Discovery (b) Enforcement of Authentication & Authorization policies (c) Network Resilience features and (d) Monitoring Capabilities.
- Deployment Guidance is needed to provide the above functions as several configuration parameters are involved spanning several service mesh components.
Document & Public Comments

• Announcement
https://csrc.nist.gov/news/2020/draft-sp-800-204a-available-for-comment

• Document Location
https://csrc.nist.gov/publications/detail/sp/800-204a/draft

• Email Comments to
sp800-204A-comments@nist.gov

• Authors
Ramaswamy Chandramouli – mouli@nist.gov
Zack Butcher - zack@tetrate.io
Istio and Envoy

Liam White
Engineer
Tetrate
Istio & Envoy: Building the Mesh

Liam White
Software Engineer, Tetrate
Story as old as time:
Service A meets service B...
Deploy a proxy (Envoy) beside your application ("sidecar deployment")
Deploy Galley to configure the rest of the Istio control plane
Deploy Pilot
to configure the sidecars
...add Mixer to enforce policy
...finally, add Citadel to assign SPIFFE identities and enable secure communication
Together we have the Istio Mesh Control Plane
Now, let’s track that call
Envoy intercepts it
and picks a new destination
the receiving Envoy checks policy
then hands the request to B
Envoy A answers Envoy B

B answers

Pilot

Mixer

Citadel
Both Envoys asynchronously report traces
and both Envoys expose metrics
**Pilot** - Control plane to configure and push service communication policies.

**Envoy** - Network proxy to intercept communication and apply policies.

**Mixer** - Policy enforcement with a flexible plugin model for providers for a policy.

**Citadel** - Service-to-service auth[n,z] using mutual TLS, with built-in identity and credential management.

**Galley** - Configuration validation, distribution
● A lightweight proxy configured with Discovery Services APIs

● xDS APIs
  ○ Route (RDS)
  ○ Cluster (CDS)
  ○ Listener (LDS)
  ○ Endpoint (EDS)
  ○ Secret (SDS)

● Push based model
Mesh Metrics

- A service mesh provides *operational* metrics
  - “RED” metrics
    - Rate
    - Errors
    - Duration
  - black box - no business metrics
- Common dimensions across all services
  - with same semantics!
- Gathered/reported directly from Envoys
Mesh Logs

Date: 2016-04-15T20:17:00.310Z

Request Info:
- "POST /api/v1/locations HTTP/2" 204
- "nsq2http"
- "cc21d9b0-cf5c-432b-8c7e-98aeb7988cd2"
- "locations"

Envoy Response Flags:
- 154 0 226 100 "10.0.35.28"

Forwarded For:
tcp://10.0.2.1:80

Upstream Host:
"10.0.35.28"
Tracing

- No magic bullet, Istio can only help
- Envoy *can*:
  - Initiate a trace
  - Propagate trace data into your application
- Envoy *cannot*:
  - Propagate data automatically onto outbound requests
Istio Security in a Slide

- Authorize access based on service identity or any channel attribute.
- Enable mTLS for authentication and encryption.
- Configure finer grained RPC-level access control for REST and gRPC.
Authentication

- Each workload is assigned a unique identity that it uses to communicate with every other workload.
- Identity is encoded as a SPIFFE X.509 SVID.
  - URI in Subject Alt Name (SAN) Field
SPIFFE

- Naming scheme for workload identities
- How to encode those names into an X.509 certificate
- How a client validates an X.509 certificate to authenticate the SPIFFE identity inside of it
SPIFFE Identity

spiffe:///trust-domain/path
Identifying Service Owner

spiffe://k8s-west.example.com/ns/staging/sa/default
Identifying Service

spiffe://staging.example.com/payments/mysql
Opaque Identity (UUID)

spiffe://example.com/9eebccd2-12bf-40a6-b262-65fe0487d453
Can You perform Action on Object?

Can Service A perform a PUT request on Service B
Summary

- Consistent metrics, tracing, access logs
- Decouple network management from your application
- Default mTLS communication everywhere (AuthN)
- Policy enforcement points at each hop (AuthZ)
Questions?

liamandrewwhite

liam@tetrate.io
Questions
(Slido)
Lunch
How did the DoD move to Kubernetes and Istio

Nicolas M. Chaillan
Chief Software Officer, Air Force
Co-lead, DoD Enterprise DevSecOps Initiative
How did the Department of Defense move to Kubernetes and Istio?

Mr. Nicolas Chaillan
Chief Software Officer, U.S. Air Force
Co-Lead, DoD Enterprise DevSecOps Initiative
Must Rapidly Adapt To Challenges
Work as a Team!
Work as a Team!

A Large Team!
Must Adapt To Challenges

Work as a Team!

A Large Team!

With Various Technologies
Must Adapt to Challenges

Work as a Team!

A Large Team!

With Various Technologies

Bring It With Us!
Even To Space!
With a Few Sensors!
Must Adapt To Challenges

Work as a Team!

A Large Team!

With Various Technologies

Bring It With Us!

To Space!

With a Few Sensors!

With Their Help!
What is the DoD Enterprise DevSecOps Initiative?

- Technology excellence in execution is enabled by
  - **Avoid vendor lock-in** at the Infrastructure and Platform Layer by leveraging FOSS with **Kubernetes and OCI containers**.
  - Creating the **DoD Centralized Artifacts Repository (DCAR)** of hardened and centrally accredited containers,
  - **Baked-in Zero Trust Security** with our **Sidebar Container Security Stack (SCSS)** leveraging **behavior detection**, **zero trust down to the container/function level**.
  - Leveraging a Scalable Microservices Architecture with **Service Mesh** baked-in security, and the adoption of automation and services for platform, infrastructure, configuration, and continuous risk assessment.
- Bringing **Enterprise IT Capabilities with Cloud One and Platform One** – Cloud and DevSecOps as Managed Services capabilities, on-boarding and support!
- Standardizing metrics and define acceptable thresholds for **DoD-wide continuous Authority to Operate**.
- Massive **Scale Training with Self Learning Capabilities** (train over 100K people within a year) and bring state of the art DevSecOps curriculum
- Creating new Agile contracting language to enable and incentivize the use of DevSecOps
From Waterfall to DevSecOps
DoD Enterprise DevSecOps Technology Stack
(Exemplar)

PLAN & DEVELOP

BUILD

TEST

SECURE

STORE ARTIFACTS

DEPLOY & OPERATE

SCALE

MONITOR

“Continuous Integration & Continuous Delivery” Orchestration

Container and Container Management

DoD Enterprise DevSecOps Technology Stack
(Exemplar)
“Cloud One” vs “Platform One by LevelUP”

- Cloud One:
  - Centralized team to provide Cloud Infrastructure with baked-in security to DoD programs. Think of it as the Infrastructure team with baked-in security, CSSP and Authority to Operate (ATO).

- Platform One by LevelUP:
  - Centralized team to provide DevSecOps/Software Factory with baked-in security to DoD Programs. Think of it as the Platform Team with the ability to deploy a DevSecOps (Kubernetes compliant) Platform and CI/CD pipeline with a Continuous ATO (c-ATO). You select from accredited tools to accelerate your ability to focus on delivering mission capabilities.
Understanding the DevSecOps Layers

- **Infrastructure Layer**: Environment Agnostic
  - Cloud One Preferred for unclassified (IL2, IL4, IL5) (AWS GovCloud/Azure Gov)
  - Or SC2S/C2S/FENCES
  - Or on-premise/classified environments

- **Platform Layer**: Fully containerized, leverages DoD approved containers from DCAR
  - Development Team selects tools from 172 approved containers or custom containers
  - CNCF compliant Kubernetes (K8S)
  - Includes Site Reliability Engineers (SREs) etc.

- **Service Mesh Layer**: Development Teams can build software/microservices leveraging hardened containers
  - Brings baked-in security and Microservices architecture enablement

- **Continuous Integration / Continuous Delivery (CI/CD) Layer**: Development Team selects between approved K8S stacks
  - Leverages the Sidecar Container Security Stack

- **Application Layer**: Team
  - One Platform
  - One Cloud

For Official Use Only - FOOU
Why Kubernetes / Containers?

- One of the most critical aspects of the DevSecOps initiative is to ensure we avoid any vendor lock-in so the DoD mandated:
  - Open Container Initiative (OCI) containers (no lock-in to containers/container runtimes/builders)
  - Cloud Native Computing Foundation (CNCF) Kubernetes compliant cluster for container orchestration, no lock-in to orchestration options/networking/storage APIs.
- Containers are immutable and will allow the DoD to centrally accredit and harden containers (FOSS, COTS, GOTS) (think of a true gold disk concept but that actually scale and works).
- Kubernetes will provide:
  - Resiliency: Self-healing so containers that crash can automatically be restarted,
  - Baked-in security: thanks to automatic injection of our Sidecar Container Security Stack (SCSS) to any K8S cluster with Zero Trust,
  - Adaptability: containers are “Lego” blocks and can be swapped with no downtime thanks to load balancing and modern routing (A/B testing, canary release etc.),
  - Automation: thanks to our Infrastructure as Code (IaC) and GitOps model,
  - Auto-scaling: if load requires more of the same container, K8S will automatically scale based on compute/memory needs,
  - Abstraction layer: ensure we don’t get locked-in to Cloud APIs or to a specific platform as K8S is managed by CNCF and dozens of products are compliant with its requirements.
Key “Continuous Security”
Ingredients

- **Kubernetes hardening.**
  - Automated injection of Sidecar Container Security Stack (SCSS) into all containers/pods running without manual action.
  - RBAC/SSO/SELinux enabled
  - Compliant with CIS Kubernetes Benchmark, mapped to NIST 800-53
  - Nodes, master, etcd are hardened.
  - Automated backups of cluster and persistent storage!

- **Sidecar Container Security Stack (SCSS):**
  - Automated centralized logging and telemetry with Elasticsearch, Fluentd, Kibana (EFK),
  - Service Mesh (Istio):
    - Baked-in [zero trust model](#) down to the container level!
      - Strong identities automatically generated using certificates.
      - mTLS tunnel injected across all container communication
    - Whitelist enforcement, Layer 7 load balancer etc.
  - Container security: Continuous Scanning, Alerting, CVE scanning, **Behavior detection** both in development and production (Build, Registry, Runtime) with Twistlock (looking into StackRox and Sysdig),
  - Container security and insider threat (custom policies detecting unapproved changes to Dockerfiles) with Anchore;
  - Automated STIG compliance with OpenSCAP.
Microservices Architecture (ISTIO)

- Turnkey Service Mesh (ISTIO) architecture
- ISTIO side car proxy, baked-in security, without any developer interaction or cod
- Benefits:
  - API Management, service discovery, aut
  - Dynamic request routing for A/B testing, observability, retries, circuit breakers and
  - Layer 7 Load balancing
  - Zero Trust model: East/West Traffic Whil
  - TLS encryption by default, Key managem
Abstracted: to avoid drifts, be agnostic to environment (Cloud/on-premise/classified/disconnected…) and prevent lock-ins with Cloud or Platform layers, we leverage CNCF compliant Kubernetes and OCI compliant containers - open source stacks with U.S eyes on code and continuous scanning,

GitOps / Infrastructure as Code (IaC): no drift, everything is code (including configuration, networking etc.) Instantiate entire stack automatically,

Continuous Integration/Continuous Delivery pipeline (CI/CD): fully containerized and using Infrastructure as Code (IaC),

Hardened Containers: hardened “Lego blocks” to bring options to development teams (one size fits all lead to shadow IT)

Software Testing: mandated high test coverage,

Baked-in Security: mandated static/dynamic code analysis, container security, bill of material (supply chain risk) etc.

Continuous Monitoring:

- Centralized logging and telemetry,
- Automated alerting,
- Zero trust, leveraging Service Mesh as Sidecar (part of SCSS), down to the container level,
- Behavior detection (automated prevention),
- CVE scanning,

Chaos engineering: Dynamically kills/restarts container with moving target defense.
The “Infrastructure as Code” concept is a critical DevSecOps ingredient to ensure that production environments do not drift from development/testing environments. No human should make changes in production environments. Changes should only be made in source code and redeployed by the CI/CD pipeline.

- No drift between environments, whether classified/disconnected/Cloud/on-premise
- Immutable,
- Replicable,
- Automated,
- No human in production environments: reduces attack surface (disable SSH etc.), insider threat and configuration drifts,
- Everything is code: including playbooks, networking, tests, configuration etc.
What is a Continuous ATO?

- A Continuous ATO is very different from a traditional ATO or a Fast-Track/Accelerated ATO:
  - Platforms have to be compliant with the DoD Enterprise DevSecOps Ref Design to ensure DoD-wide reciprocity, including the use of the Sidecar Container Security Stack (SCSS). Platform controls are mapped to NIST-800-53.
  - We accredit the Platform’s **PROCESS** (Continuous Integration/Continuous Delivery (Software Factory)) with mandated **testing and security gates**. The software coming out of the factory and that is RUNNING IN PRODUCTION on the **Platform** (Kubernetes with SCSS) also benefits from the cATO.
  - We accredit **TEAMS** using the Platform so they can produce quality software and be trained to move to DevSecOps
  - A key principle of DevSecOps is the **baked-in security** with:
    - Zero Trust
    - Automation
    - Removal of environment drifts
    - Behavior Detection
    - Continuous Monitoring
    - Pen-testing

---

For Official Use Only - FOUO

**Integrity - Service - Excellence**
DevSecOps Stack implements Zero Trust!

- **Identities:**
  - strong NPE identities are automatically managed by Istio (Service Mesh) for each container to enable zero trust down to the container level.
  - Non-NPE identities are using strong identities with DoD PKI

- **Devices:**
  - Developer endpoints are using VDI options or approved endpoints images

- **Applications:**
  - Apps are containerized and behind the Service Mesh which enforces Zero trust with strong identities per pod/container and.

- **Infrastructure:**
  - Kubernetes is centrally hardened and continuously monitored with centralized logs and telemetry.
  - SCSS monitors container signatures and container state
  - SCSS brings Behavior detection and CVE continuous scanning

- **Network:**
  - mTLS tunnels are automatically injected across all containers/pods by SCSS.

- **Data:**
  - Data is always encrypted in transit and leverages FIPS encryption at rest.
CSO Website – Continuously Updated!

- Want to find information about the DevSecOps initiative and the CSO?
  - [https://software.af.mil/](https://software.af.mil/)
  - **Our latest documents/videos:** [https://software.af.mil/dsop/documents/](https://software.af.mil/dsop/documents/)
  - **Our latest training videos from DAU available at:** [https://software.af.mil/training/](https://software.af.mil/training/)

- More information about
  - Cloud One
  - Platform One
  - DevSecOps
  - Training including videos selection
  - Software Factories
  - Our Events/News!
Thank You!

Nicolas Chaillan
Chief Software Officer, U.S. Air Force
usaf.cso@mail.mil
Backup Slides
Nicolas M. Chaillan is the Chief Software Officer at the U.S. Air Force and the Co-Lead for the DoD Enterprise DevSecOps Initiative.

He is the former Special Advisor for Cloud Security and DevSecOps at OSD, A&S.

He was the Special Advisor for Cybersecurity at the Department of Homeland Security and the Chief Architect for Cyber.gov, the new robust, innovative and holistic .Gov cyber security architecture for all .gov agencies.

Chaillan is a technology entrepreneur, software developer, cyber expert and inventor. He is recognized as one of France’s youngest entrepreneurs after founding his first company at 15 years of age.

With 19 years of international tech, entrepreneurial and management experience, Chaillan is the founder of more than 12 companies, including AFTER-MOUSE.COM, Prevent-Breach, anyGuest.com, and more.

Over the last eight years alone, he has created and sold over 180 innovative software products to 40 Fortune 500 companies.

Chaillan is recognized as a pioneer of the computer language PHP.
DCCSCR/DCAR
(DoD Container Repository)

- Containers are centrally accredited by the DSOP team in the DoD repository:
  - **DoD Centralized Containers Source Code Repository (DCCSCR):** https://dccscr.dsop.io/dsop
    - Allows DoD programs to reuse DevSecOps stack and CI/CD pipelines to ensure pre-hardened deployments.
  - **DoD Centralized Artifacts Repository (DCAR) (Container binaries):** https://dcar.dsop.io
- Containers are signed and continuously monitored.
- Community can contribute code merge requests, reviewed by the DSOP team.
- Vendors/DoD Programs can contribute containers that have enterprise benefits to DCCSCR/DCAR and DSOP team will accredit them and maintain them.
Value for DoD Programs

- Enables any DoD Program across DoD Services deploy a DoD hardened Software Factory, on their existing or new environments (including classified, disconnected and Clouds), within **days instead of a year**. Tremendous cost and time savings.
- Multiple DevSecOps pipelines are available with various options (no one-size-fits-all)
- Enables rapid prototyping (in days and not months or years) for any Business, C4ISR and Weapons system. Deployment in PRODUCTION!
- Enables learning and **continuous feedback** from actual end-users (warfighters).
- Enables **bug and security fixes in minutes** instead of weeks/months.
- Enables automated testing and security.
- Enables **continuous Authorization to Operate (c-ATO)** process. Authorize ONCE, use MANY times!
- Brings a holistic and baked-in cybersecurity stack, gaining complete visibility of all assets, software security state and infrastructure as code.
Air Force Cloud Office with turnkey access to AWS GovCloud and Azure Government at IL2, 4 and 5. IL6 available by December 2019.

Simple “Pay per use” model with ability to instantiate your own Development and Production VPCs at various Impact Levels within days with full compliance/security and a baked-in ATO.

Enterprise Solution: we provide the guardrails to the cloud in a standard manner so you can focus on your mission

Fully Automated: All environmental stand-up is managed by Infrastructure as Code, drastically speeding up deployment, reducing manual work, and human error

Centralized Identities and Single-Sign-On (SSO): one login across the Cloud stack

Internet facing Cloud based VPN to connect to IL5 enclaves with a Virtual Internet Access Point (coming within January 2020).

DevSecOps Focused: secure, mission driven deployments are built into the framework to ensure self-service and seamless deployments. Leverages Zero Trust model.

Proactive Scaling and System Monitoring: Mission Owners can see all operational metrics and provide rules and alerts to manage each mission their way

Accreditation Inheritance has been identified in the AF-Cloud One eMASS accounts (AWS & Azure) to include inheritance from the CSP, USAF, DoD and CSSP. All that’s left for the mission is the controls that are unique to them.
“Platform One by LevelUP”
The Air Force Software Factory Team

- Merged top talent across U.S. Air Force from various Factories (Kessel Run, SpaceCAMP and UP).
- Helps instantiate DevSecOps CI/CD pipelines / Software Factories within days at various classification levels.
- Manages Software Factories for Development teams so they can focus on building mission applications.
- Provides Blanket Purchase Agreement (BPA) DoD-wide DevSecOps contracts for Cloud Service, Talent and Licenses. Enables awards every 15/30 days with bulk discounts.
- Decouples Development Teams from Factory teams with DevSecOps and Site Reliability Engineer (SRE) expertise.
- Partners with Cloud One to provide IL2, 4, 5 and 6 access but also uses C2S/SC2S and various on-premise environments!
- Self-learning and training capabilities to enable teams move to Scrum/Kanban/eXtreme Programming (XP) Agile practices.
- Leverages the DoD hardened containers while avoiding one-size-fits-all architectures.
- Fully compliant with the DoD Enterprise DevSecOps Initiative (DSOP) with DoD-wide reciprocity and an ATO. Leverages Zero Trust model.
- Hardens the 172 DoD enterprise containers (databases, development tools, CI/CD tools, cybersecurity tools etc.).
- Provides Software Enterprise Services with Collaboration tools, Cybersecurity tools, Source code repositories, Artifact repositories, Development tools, DevSecOps as a Service, Chats etc. These services will be MANAGED services on Cloud One.
- Hardened Containers Options
  - Delivery of hardened enterprise containers with accreditation reciprocity (existing containers only).
  - Delivery of custom hardened containers as needed.

- Continuous Integration / Continuous Delivery (CI/CD) Options
  - Delivery of existing hardened Kubernetes/OpenShift/PKS playbooks (full Infrastructure as Code).
  - Delivery of a **turnkey CI/CD pipeline** (Software Factory) with complete « Infrastructure as Code » to instantiate on any environment (development teams picks the tools from the approved hardened containers) on various classified/unclassified environment.

- Training/On-Boarding Options
  - 1-day training Session: introduction to DevSecOps. Overview and understanding of the vision and activities.
  - A 3 day introduction to LevelUP DevSecOps tech stack. Hands on code and User-Centered Design (UCD) to deploy your first demo app to production.
  - A several week full on-boarding, that concludes with an MVP ready for production.
  - A several month full on-boarding, that concludes with your platform team being able to support your own DevSecOps applications for development and production.
  - Customized training options (both at our locations or on your premises).

- Contracting Support Options
  - Ability to leverage the DevSecOps BOAs (Cloud Services, Talent and Licenses).
  - Enable access to DevSecOps engineers/SREs Full-Time-Equivalent (FTEs) (Medics/Counselors) to assist Programs.
DoD Enterprise DevSecOps Architecture
DoD Enterprise DevSecOps Architecture*

*each DoD Program can have its own instantiation of the DoD Enterprise DevSecOps Platform on any Cloud.

** can be installed with single command and deployed on any Cloud.

*** could be deployed inside an enclave or on-premises

**** gives complete visibilities of assets, security/vulnerability state etc. can be integrated to existing cybersecurity shared services.

For Official Use Only - FOOU

I n t e g r i t y  -  S e r v i c e  -  E x c e l l e n c e
DevSecOps Platform Stack (continuously evolving)
### DevSecOps Product Stack (1)

<table>
<thead>
<tr>
<th>Source Repository</th>
<th>API Gateways</th>
<th>Programming Languages</th>
<th>Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub Government</td>
<td>Kong</td>
<td>C/C++</td>
<td>SQL Server</td>
</tr>
<tr>
<td>GitLab</td>
<td>Azure API</td>
<td>C#.NET</td>
<td>MySQL</td>
</tr>
<tr>
<td></td>
<td>AWS API</td>
<td>.NET Core</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>Axway</td>
<td>Java</td>
<td>MongoDB</td>
</tr>
<tr>
<td></td>
<td>3Scale</td>
<td>PHP</td>
<td>SQLite</td>
</tr>
<tr>
<td></td>
<td>Apigee</td>
<td>Python</td>
<td>Redis</td>
</tr>
<tr>
<td></td>
<td>ISTIO (service mesh)</td>
<td>Groovy</td>
<td>Elasticsearch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ruby</td>
<td>Oracle</td>
</tr>
<tr>
<td><strong>Container Management technologies:</strong></td>
<td><strong>Artifacts</strong></td>
<td><strong>R</strong></td>
<td>etcd</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>Artifactory</td>
<td><strong>Rust</strong></td>
<td>Hadoop/HDInsight</td>
</tr>
<tr>
<td>Openshift</td>
<td>Nexus</td>
<td><strong>Scala</strong></td>
<td>Cloudera</td>
</tr>
<tr>
<td>VMWare Tanzu</td>
<td>Maven</td>
<td><strong>Perl</strong></td>
<td>Oracle Big Data</td>
</tr>
<tr>
<td>PKS</td>
<td>Archiva</td>
<td><strong>Go</strong></td>
<td>Solr</td>
</tr>
<tr>
<td>OKD</td>
<td>S3 bucket</td>
<td><strong>Node.JS</strong></td>
<td>Neo4J</td>
</tr>
<tr>
<td>D2IQ (K8S only)</td>
<td></td>
<td><strong>Swift</strong></td>
<td>Memcached</td>
</tr>
<tr>
<td>Docker EE (K8S only)</td>
<td></td>
<td></td>
<td>Cassandra</td>
</tr>
<tr>
<td><strong>Container Packagers:</strong></td>
<td><strong>Artifacts</strong></td>
<td></td>
<td>MariaDB</td>
</tr>
<tr>
<td>Helm</td>
<td>Artifactory</td>
<td></td>
<td>CouchDB</td>
</tr>
<tr>
<td>Kubernetes Operators</td>
<td>Nexus</td>
<td></td>
<td>InfluxDB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(time)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programming Languages</th>
<th>Databases</th>
<th>Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>C/C++</td>
<td>SQL Server</td>
<td></td>
</tr>
<tr>
<td>C#.NET</td>
<td>MySQL</td>
<td></td>
</tr>
<tr>
<td>.NET Core</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>MongoDB</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>SQLite</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>Redis</td>
<td></td>
</tr>
<tr>
<td>Groovy</td>
<td>Elasticsearch</td>
<td></td>
</tr>
<tr>
<td>Ruby</td>
<td>Oracle</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>etcd</td>
<td></td>
</tr>
<tr>
<td>Rust</td>
<td>Hadoop/HDInsight</td>
<td></td>
</tr>
<tr>
<td>Scala</td>
<td>Cloudera</td>
<td></td>
</tr>
<tr>
<td>Perl</td>
<td>Oracle Big Data</td>
<td></td>
</tr>
<tr>
<td>Go</td>
<td>Solr</td>
<td></td>
</tr>
<tr>
<td>Node.JS</td>
<td>Neo4J</td>
<td></td>
</tr>
<tr>
<td>Swift</td>
<td>Memcached</td>
<td></td>
</tr>
</tbody>
</table>

**Languages:**
- C/C++
- C#.NET
- .NET Core
- Java
- PHP
- Python
- Groovy
- Ruby
- R
- Rust
- Scala
- Perl
- Go
- Node.JS
- Swift

**Databases:**
- SQL Server
- MySQL
- PostgreSQL
- MongoDB
- SQLite
- Redis
- Elasticsearch
- Oracle
- etcd
- Hadoop/HDInsight
- Cloudera
- Oracle Big Data
- Solr
- Neo4J
- Memcached
- Cassandra
- MariaDB
- CouchDB
- InfluxDB (time)
# DevSecOps Product Stack (2)

<table>
<thead>
<tr>
<th>Message bus/Streams</th>
<th>Logs</th>
<th>Docker base images OS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kafka</td>
<td>Logstash</td>
<td>Alpine</td>
</tr>
<tr>
<td>Flink</td>
<td>Splunk Forwarder</td>
<td>Busybox</td>
</tr>
<tr>
<td>Nats</td>
<td>Fluentd</td>
<td>Ubuntu</td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>Syslogd</td>
<td>CentOS</td>
</tr>
<tr>
<td>ActiveMQ</td>
<td>Facebeat</td>
<td>Debian</td>
</tr>
<tr>
<td></td>
<td>rsyslog</td>
<td>Fedora</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Universal Base Image</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proxy</th>
<th>Websevers</th>
<th>Serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oauth2 proxy</td>
<td>Apache2</td>
<td>Knative</td>
</tr>
<tr>
<td>nginx ldap auth proxy</td>
<td>Nginx</td>
<td></td>
</tr>
<tr>
<td>openldap</td>
<td>IIS</td>
<td></td>
</tr>
<tr>
<td>HA Proxy</td>
<td>Lighttpd</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tomcat</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visualization</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tableau</td>
<td></td>
</tr>
<tr>
<td>Kibana</td>
<td></td>
</tr>
</tbody>
</table>
## DevSecOps Product Stack (3)

<table>
<thead>
<tr>
<th>Build</th>
<th>Test coverage</th>
<th>Security</th>
<th>Security (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSBuild</td>
<td>JaCoCo</td>
<td>Tenable / Nessus Agents</td>
<td>Snyk</td>
</tr>
<tr>
<td>CMake</td>
<td>Emma</td>
<td>Fortify</td>
<td>Code Climate</td>
</tr>
<tr>
<td>Maven</td>
<td>Cobertura</td>
<td>Twistlock</td>
<td>AJAX Spider</td>
</tr>
<tr>
<td>Gradle</td>
<td>codecov</td>
<td>Aqua</td>
<td>Tanaguru (508 compliance)</td>
</tr>
<tr>
<td>Apache Ant</td>
<td></td>
<td>SonarQBE</td>
<td>InSpec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Qualys</td>
<td>OWASP Dependency-Check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StackRox</td>
<td>Burp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aporeto</td>
<td>HBSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Snort</td>
<td>Anchore</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OWASP ZAP</td>
<td>Checkmarx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contrast Security</td>
<td>SD Elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OpenVAS</td>
<td>Clair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Metaspliot</td>
<td>Docker Bench Security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ThreadFix</td>
<td>Notary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pylint</td>
<td>Sysdig</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JFrog Xray</td>
<td>Layered Insight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OpenSCAP (can check against</td>
<td>BlackDuck</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISA STIG</td>
<td>Nexus IQ/Lifecycle/Firewall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OpenControl for compliance</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>documentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CI/CD Orchestration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jenkins (open source)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CloudBees Jenkins</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GitLab</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jenkins plugins</td>
<td>Dozens (Need to verify security).</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configuration Management / Delivery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chef</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ansible</td>
<td>Saltstack</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tenable / Nessus Agents</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fortify</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Twistlock</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aqua</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SonarQBE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Qualys</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>StackRox</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aporeto</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Snort</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OWASP ZAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Contrast Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenVAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Metaspliot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ThreadFix</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pylint</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JFrog Xray</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenSCAP (can check against</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DISA STIG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenControl for compliance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>documentation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tests suite:**
- Cucumber
- J-Unit
- Selenium
- TestingWhiz
- Watir
- Sahi
- Zephyr
- Vagrant
- AppVerify
- nosetests
- SoapUI
- LeanFT

**Security:**
- Tenable / Nessus Agents
- Fortify
- Twistlock
- Aqua
- SonarQBE
- Qualys
- StackRox
- Aporeto
- Snort
- OWASP ZAP
- Contrast Security
- OpenVAS
- Metaspliot
- ThreadFix
- pylint
- JFrog Xray
- OpenSCAP (can check against DISA STIG)
- OpenControl for compliance documentation

**Security (2):**
- Snyk
- Code Climate
- AJAX Spider
- Tanaguru (508 compliance)
- InSpec
- OWASP Dependency-Check
- Burp
- HBSS
- Anchore
- Checkmarx
- SD Elements
- Clair
- Docker Bench Security
- Notary
- Sysdig
- Layered Insight
- BlackDuck
- Nexus IQ/Lifecycle/Firewall
## DevSecOps Product Stack (4)

### Monitoring
- Sensu
- EFK (Elasticsearch, Fluentd, Kibana)
- Splunk
- Nagios
- New Relic
- Sentry
- Prometheus
- Grafana
- Kiali

### Collaboration
- Rocket.Chat
- Matter.Most
- PagerDuty

### Plan
- Jira
- Confluence
- Rally
- Redmine
- Pivotal Tracker

### Secrets
- Kubernetes Secrets
- Vault
- Credentials (Jenkins)
- CryptoMove

### SSO
- Keycloak

### Documentation
- Javadoc
- RDoc
- Sphinx
- Doxygen
- Cucumber
- phpDocumentator
- Pydoc

### Performance
- Apache AB
- Jmeter
- LoadRunner
Recommended Videos (Part 1)

- Watch our playlists, available at different expertise levels and continuously augmented!
- Kafka / KSQL (message bus, pub/sub, event driven):
  - Beginners: [https://www.youtube.com/playlist?list=PLSIv_F9TtLizz0zt03Ludtid7icrXBesg](https://www.youtube.com/playlist?list=PLSIv_F9TtLizz0zt03Ludtid7icrXBesg)
  - Intermediate: [https://www.youtube.com/playlist?list=PLSIv_F9TtLxxXX0oCzt7laO6mD61UIQw](https://www.youtube.com/playlist?list=PLSIv_F9TtLxxXX0oCzt7laO6mD61UIQw)
  - Advanced: N/A

- Kubernetes
  - Beginners: [https://www.youtube.com/playlist?list=PLSIv_F9TtLlydFzQzkYYDdQK7k5cEKubQ](https://www.youtube.com/playlist?list=PLSIv_F9TtLlydFzQzkYYDdQK7k5cEKubQ)
  - Intermediate: [https://www.youtube.com/playlist?list=PLSIv_F9TtLlx8dSFH_jFLK40Tt7KUXTN](https://www.youtube.com/playlist?list=PLSIv_F9TtLlx8dSFH_jFLK40Tt7KUXTN)
  - Advanced: [https://www.youtube.com/playlist?list=PLSIv_F9TtLlytdAJiVqbHucWOvn5LrTNW](https://www.youtube.com/playlist?list=PLSIv_F9TtLlytdAJiVqbHucWOvn5LrTNW)
Self-Learning (2)

- **Recommended Videos (Part 2)**
  - Watch our playlists, available at different expertise levels and continuously augmented!
  - Service Mesh
    - Beginners: https://www.youtube.com/playlist?list=PLSlv_F9TtLixtC4rDIMQ8QiG5UbCjz7VH
    - Intermediate: https://www.youtube.com/playlist?list=PLSlv_F9TtLiwWK_Y_Cas8Nyw-Dsdh6vL
    - Advanced: https://www.youtube.com/playlist?list=PLSlv_F9TtLix8VW2MFONMRws_-2rSJw7dn
  - Microservices
    - Beginners: https://www.youtube.com/playlist?list=PLSlv_F9TtLi0z_U2_RaONTGYLkJz0h-A_L
    - Intermediate: https://www.youtube.com/playlist?list=PLSlv_F9TtLxqjuAXoRMjvspaEE8L2cB
    - Advanced: https://www.youtube.com/playlist?list=PLSlv_F9TtLw4CF4F4t3gVV3j0512CMsu
Recommended Books

- A Seat at the Table – by Mark Schwartz (former CIO of USCIS, leader in Agile)
  This book is highly recommended for ALL leadership as it is not technical but focused on the challenges around business, procurement and how leadership can enable DevOps across the organization and remove impediments.
- The Phoenix Project – by the founders of DevOps

For those who drive to work like me (for hours), please note that these books are available as Audiobooks.
Martin Fowler describes the **Strangler Application**:  

One of the natural wonders of this area are the huge strangler vines. They seed in the upper branches of a fig tree and gradually work their way down the tree until they root in the soil. Over many years they grow into fantastic and beautiful shapes, meanwhile strangling and killing the tree that was their host.

To get there, the following steps were followed:  

- First, add a proxy, which sits between the legacy application and the user. Initially, this proxy doesn’t do anything but pass all traffic, unmodified, to the application.
- Then, add new service (with its own database(s) and other supporting infrastructure) and link it to the proxy. Implement the first new page in this service. Then allow the proxy to serve traffic to that page (see below).
- Add more pages, more functionality and potentially more services. Open up the proxy to the new pages and services. Repeat until all required functionality is handled by the new stack.
- The monolith no longer serves traffic and can be switched off.

Demo 1:
Tetrate Q + Istio for Service-to-Service Access Control

Demo 2:
ABAC for Files and DBMS Elements in Multi-cloud
Tetrate
The service mesh company
Traditional access control for databases is provided by network reachability and DB credentials
If an attacker breaks into the system and gains access to the network, the data is compromised.
A service mesh provides proper Identity primitives to enforce runtime authentication.
It also provides authorization primitives to be enforced at runtime.
Access decisions can be made based on proper identity and high level concepts
EXAMPLE
Unauthorized access
NGAC provides a **context-ful** authorization framework
L7 policies can be enforced, because the proxies understand L7 protocols.
EXAMPLE
L7 policy enforcement
L7 policies can be enforced, because the proxies understand L7 protocols
This context-ful framework can be used to enforce complex and dynamic policies that are environment-dependent.
EXAMPLE
Policy combination:
RBAC + Location + Time
Thanks
Administering Access Control Logic

Ulrich Lang
ObjectSecurity

Jayanta Kumar Debnath
InfoBeyond Technology

2:30 pm

National Institute of Standards and Technology
U.S. Department of Commerce
Administering Access Control Logic

- Security Policy Test tool for Access Control Leakage Detection

Jayanta Debnath, InfoBeyond Technology LLC
Email: jayanta@infobeyondtech.com
Why Security Policy Tool?

AC policies composed and deployed into an AC system without comprehensive security tests and verifications might contain unintentional Access Control Flaws.

Access control flaws caused by misconfigured rules, policies, or algorithms within an access control model.

Systematic Policy Testing & Verification

Verified Security Policy Model

Access Control Flaws:
- Block Privilege
- Leak Privilege
- Not Protected Resource
- Rule Conflict
- Inconsistent Assignment
- Inheritance Loop
- Undecided Rules
- Separation of Duty

Security Policy Tool delivers a solution for testing, analyzing, and editing access control policies to prevent Access Control Flaws.
Security Policy Tool (SPT) Overview

SPT includes four major components:

- Policy Modeling
- Static Verification
- Dynamic Verification
- Policy Implementation

Static Verification:
- Merged Policy Verification
- Combined Policy Verification

Dynamic Verification:
- Merged Policy Verification
- Combined Policy Verification

Policy Modeling:
- ABAC, Multi-level, Workflow etc.

Policy Implementation:
- XACML conversion
- XACML editing

Attributes

Inheritance Relations

Policy Rules

Access Control Policies

XACML Policies
Policy Modeling

1. Define Attribute(s) (i.e., Subject, Resource, Action and Environment) and Condition.

2. Define Subject and Resource Inheritances

3. Define Policy

4. Add Rules Under the Policy

5. Policy Summary

Access Control Models:
- ABAC
- Multilevel
- Workflow
Policy Test and Analysis

- Security Policy Tool converts a policy to a FSM Model.
- A symbolic model checker (i.e., NuSMV), using state space search, verify whether a security requirement is true or false.

- Policy/Rule Combination Algorithms:
  - First Applicable
  - Deny Override
  - Permit Override
  - Only One Applicable
  - Deny Unless Permit
  - Permit Unless Deny
  - Weak Consensus
  - String Consensus
  - Weak Majority
  - Strong Majority
  - Super-Majority-Permit

- Policy Enforcement Algorithms:
  - Deny Biased
  - Permit Biased
Merged Policy Verification

(1) Security Requirement is evaluated with each rule

(2) Results are merged by Rule Combining Algorithm

(3) Enforced as one policy
Policy Verification Demonstration

The Security Requirement is verified against each rules from all policies individually and conflicts are resolved using rule combination algorithm if multiple rules matching the security requirement.
Combined Policy Verification

1. Security Requirement (Subject, Resource, Action, Environment, Condition)
   - Policy 1: (rule_{1,1}, rule_{1,2}, ..., rule_{1,A})
   - Policy 2: (rule_{2,1}, rule_{2,2}, ..., rule_{2,B})
   - ... 
   - Policy n: (rule_{n,1}, rule_{n,2}, ..., rule_{n,L})

2. Policy enforcement to combine the result

   (1) Security Requirement is evaluated with policy

   Permit/Deny/NA

   Permit/Deny

   Permit/Deny/NA

   Permit/Deny/NA

   Permit/Deny/NA

   Permit/Deny

   TRUE/FALSE

   Permit/Deny
Policy Verification Demonstration

Combined Policy Verification Result of Highlighted Security Requirement

Policy Combined Result: Deny

Matching rule from Medical Policy

Matching rule from Manager Policy
Separation of Duty Verification

Merged Policy Separation of Duty

Combined Policy Separation of Duty
Access Privilege Preview

(a): Subject Access Privilege

(b): Resource Access Privilege
XACML Policy Implementation

Valid XACML Element Suggestions

Full Reference Implementation of Both XACML 2.0 and XACML 3.0 Specification.

Automatic XACML Conversion and Interactive Editing

Dynamic Syntax Error Checking

Dropdown Menu on XACML Tree

Suggestions while typing

Hierarchical Representation of the XACML for each editing.

GUI based XACML Generation

Interactive Suggestions
Powerful, manageable, and reliable access control policy implementation with ObjectSecurity® OpenPMF™ 4.0

Ulrich Lang, PhD
CEO, ObjectSecurity LLC

Security Policy Automation → OpenPMF 4.0
Supply Chain Risk Analysis Automation
Vulnerability Assessment & Pen-Testing Automation
What most orgs want
Powerful yet effortless policy implementation that can be trusted

Access Control Policy Implementation remains challenging for many

Insufficient if manageable

Too difficult if adequate
Powerful access policies?

ABAC example

“Nurses can only access medical records of patients” + “...whose current treating physician is the same physician who the nurse is currently assigned to assist, and only if the nurse is currently badged into the same physical building as the one the patient is”

Very hard to test such a complex, dynamic policy!

P.S. “70% of organizations forecasted to use ABAC in 2020” (Gartner)
Lots of access control R&D at ObjectSecurity and elsewhere over the years...

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>access policy contract R&amp;D, invent OpenPMF concept</td>
</tr>
<tr>
<td>2002</td>
<td>start EU funded R&amp;D around OpenPMF</td>
</tr>
<tr>
<td>2003</td>
<td>centralized policy manager (technical rule), ingest and distributed enforcement across specific techs/layers</td>
</tr>
<tr>
<td>2007</td>
<td>generate technical rules from generic, intuitive policies using “model-driven security”, IDE integration etc.</td>
</tr>
<tr>
<td>2007</td>
<td>core patent: model-driven security policy automation (10+ today)</td>
</tr>
<tr>
<td>2008</td>
<td>core patent: model-driven security auditing automation</td>
</tr>
<tr>
<td>2008</td>
<td>awards: Gartner Cool Vendor, Global Security Challenge, TeleTrust Innovation Award, …</td>
</tr>
<tr>
<td>2008</td>
<td>start OpenPMF related SBIR subcontracts</td>
</tr>
<tr>
<td>2010</td>
<td>SaaS, BPMN, XACML, Web/SOA, data-centric, monitoring etc.</td>
</tr>
<tr>
<td>2016</td>
<td>NIST SBIR Phase II 9.03.02.77-R ACPT (access policy testing)</td>
</tr>
<tr>
<td>2017</td>
<td>flexibly ingest functional &amp; policy information (customizable), author using intuitive web GUI, generate technical access rules/configs for many techs/layers in-depth (customizable), test policies using formal methods, monitor, …</td>
</tr>
<tr>
<td>Current</td>
<td>ongoing R&amp;D &amp; improvements.</td>
</tr>
</tbody>
</table>
OpenPMF™ 4.0

During the SBIR, we integrated NIST ACPT research results into OpenPMF to create seamless or standalone model-driven testing features.

The difference between A-ACPTS vs ACPT for the 14-node testbed application is a significant time saving (288-307 minutes of work saved, 283-307 minutes total time saved from start to end), and improved confidence in correctness of test results.
OpenPMF™ 4.0 Security Policy Automation™

1. Import information about systems, applications, data flows, users, alerts etc.
2. Import existing security policies
3. Author security policies intuitively
4. Generate matching technical enforcement rules & configurations, incl. ABAC
5. Test & document policies using formal methods
6. Enforce consistent “defense in depth” via OpenPMF policy enforcement points and 3rd party security features/tools
7. Monitor policy enforcement centrally
8. Update & customize automatically if your IT landscape changes
OpenPMF 4.0 Security Policy Auditor

1. Import information
2. Author test requirements
3. Test & audit security policies
4. Visualize & document results
SBIR scenario testbed (Raspi cluster + AWS)

140,000 tcpdump log entries + DDS + “network XML”
OpenPMF Workflow Automation

(automates the upload → import → calculate → generate → download roundtrip)
Calculate data queries

(DDS user traffic only – not discovery traffic!)
Example high-level policy

“Only whitelist DDS app traffic in-depth based on IP/port whitelists, but not DDS chatter” – trivial example!
Example: Generate machine-enforceable rules

Demo outputs:
- DDS Permissions XML,
- OS firewall (Linux) scripts,
- OS firewall (Windows) scripts
Testing Example: high-level & low-level test properties
Testing Example: under the hood
Thank you!

Ulrich Lang, PhD
CEO, ObjectSecurity LLC
ulrich.lang@objectsecurity.com

https://objectsecurity.com/nist
https://objectsecurity.com/openpmf
https://objectsecurity.com/openpmf-auditor/
Questions
(Slido)
Break

Back @ 4:00 PM

4:00 pm

National Institute of Standards and Technology
U.S. Department of Commerce
Panel:
What are the greatest identity and access control issues and opportunities facing the cloud going forward?

Moderator: Vincent Hu
Senior Scientist
NIST

Panelist: Sandi Roddy
Chief Scientist
Cyber Warfare Systems
JHU/APL

Panelist: Sarbari Gupta
President
Electrosoft

Panelist: Ron Turner
XpressRules

Panelist: Ramaswamy Chandramouli
Senior Scientist
NIST

Panelist: Mike Fisk
Deputy CISO
Capital One

4:15 pm
Thank you

Day 1
Closing Remarks and Adjoum

5:00 pm

National Institute of Standards and Technology
U.S. Department of Commerce