Observation & Orientation for Multi-Domain Operations with an Intelligent Tactical Fog

Presented by:

MultiPlex | studio

Basil Security

Technica®

Multplex.studio was formed to tackle the AFWERX MDO Challenge and is the partnership between several independents, Technica and Basil Security. Technica brings their SmartFog platform to the equation which will be used as the backbone of this proposal and Basil Security's distributed policy enforcement system will be leveraged for multi-level classification, as well as for multi-level encryption.

The Team:

Jason Lind
Project Lead
lind@multiplex.studio

Nick Chadwick
Lead Architect
chadwick@multiplex.studio

Mitchell Maddox
Project Manager
maddox@multiplex.studio

Karl Walinskas
IR&D Business Development
kwalinskas.ctr@technicacorp.com

Brendon Unland
SmartFog Architect
bunland@technicacorp.com

Ron Herardian
Security Architect
ron@basilsecurity.com

Executive Summary

Multiplex.studio is an amalgamation of technology companies presenting an innovative, best-of-breed solution for the AFWERX MDO challenge. This proposal shows the integration of **Technica Corporation's** *SmartFog* platform technology in conjunction with **Basil Security** to create a horizontal system-level architecture to distribute computing, storage and networking functions. We provide an advanced, policy-as-code framework for distributed security policy enforcement that combines next-generation NIST attribute Based Access Control with event-driven and stateful security policy conditions.

Technica's approach with SmartFog platform consists of a system level architect that distributes computing and network functions over a Cloud-to-Thing (CtT) environment as an extension of the existing cloud-based model already in use with network topology. By implementing this strategy closer to the edge of where these Internet of Things (IoT) reside, we can process data in near real-time, validate and analyze at both the Edge and in the cloud while minimizing effects of bandwidth limitations. The Edge uses a dedicated server collecting localized data, thus reducing the burden from the core on-premise data centers within cloud systems and allowing more accurate analysis locally, so that only relevant information is then processed and sent forward. The cloud will offer more powerful computing resources and more advanced analysis to be sure; however, typical of tactical situations, communications are often denied, degraded, intermittent or limited (D-DIL) at best, driving the requirement for edge capabilities.

Basil Security offers an advanced distributed security policy enforcement effort that combines next-generation NIST attribute-based access control (ABAC) with event-driven security policy conditions. These event-driven conditions are information on the network that generates a security policy decision communicated to the policy validation system based on a change in the external environment. Basil security policies are referred to as "Smart Policies" because, unlike static access control rules, Basil policies adapt automatically under changing conditions, based on events or external state information. In this MDO proposal, Basil addresses many of the integrated information security classification issues and the technical operations for security and provides an overlay of security policies over otherwise disparate systems. Basil provides single-pane-of-glass visibility into policies across the complete, integrated MDO system.

Multiplex.studio's solution for the data architecture will be to collect data at the Edge, perform validation and analysis and securely stream those results to the cloud. Technica's SmartFog platform is leveraged for enabling seamless D-DIL streams from the fog layer to the cloud and Basil's policy enforcement engine enables multi-level classifications and data encryption in support of warfighter needs.

Our Interpretation of MDO

In today's conflicts and future wars, the ability to maintain a persistent presence, ensure a cohesive operational perspective, and maximize processing of systems and personnel is critical to effectively employ multi-domain capabilities and produce significant and simultaneous challenges for our competitors across all domains.

More than just operating weapons systems in multiple domains, Multi-Domain Operations (MDO) describes the seamless, dynamic and continuous generation of offensive/defensive effects across all the established domains: cyber, air, space, land, sea, sub-surface and electromagnetic spectrum. This includes coordination and execution of not only all five branches of the US Armed Services but Allies as well. Coalition forces must have the operational agility to seamlessly shift between levels of war, geographic areas, missions, functions and warfighting domains and must be able to dynamically create multiple dilemmas for an adversary at an operations tempo they cannot match.

MDO Strategy

The vision of MDO is one of Rapid, Resilient and Agile Adaptation and Intuition.

Rapid: from signal to decision in less than 15 minutes

Resilient: supports a dynamic battlespace and continues to operate even under attack at the edge

Agile: integrate a diverse set of warfighting players and data sources

Adaptation: must operate in a wide variety of contexts with minimum effort to reconfigure

Intuition: advanced data analysis through artificial intelligence (AI) and machine learning (ML)

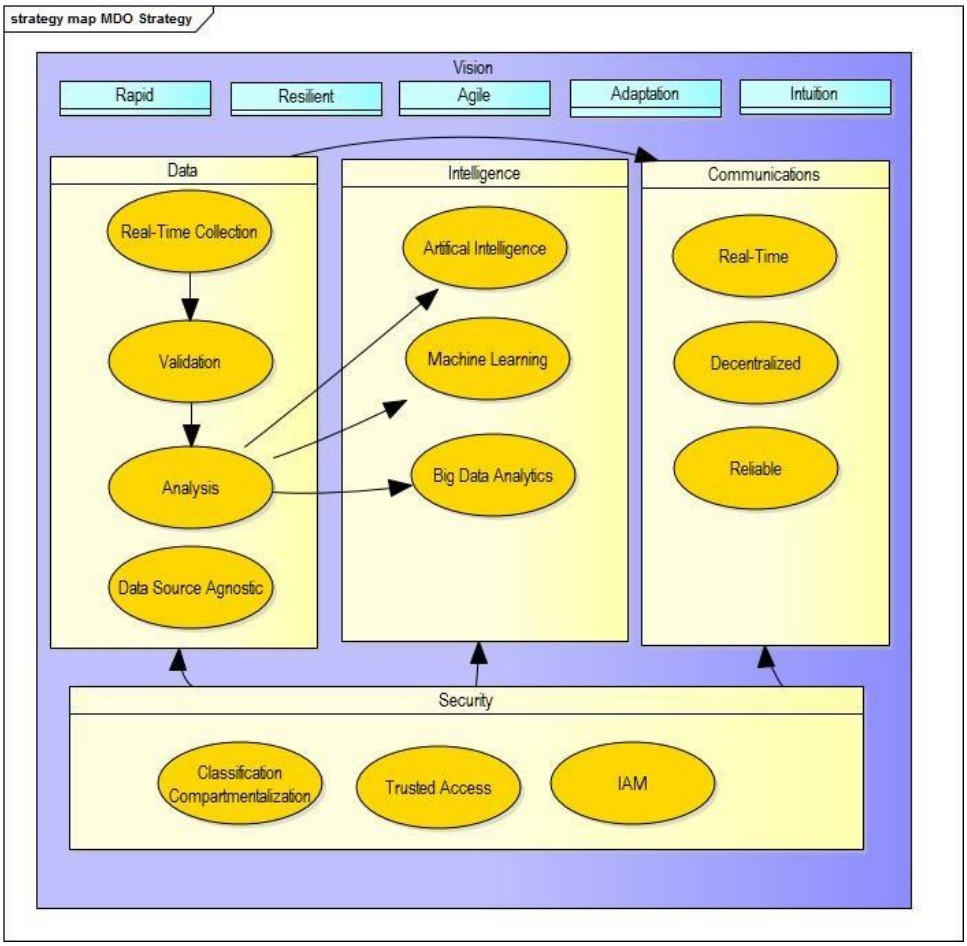

Figure 1 – MDO Strategy representation

The MDO Problem Set

The 2018 National Defense Strategy (NDS) states we cannot expect successful operations in tomorrow's conflicts with yesterday's weapons, equipment, or force structure. Currently, the Air Force does not adequately integrate, fuse, plan, employ or debrief information (e.g. intelligence) and effects across all domains. In future multi-domain battlespaces, unassured communications, and siloed information and capabilities (due to lack of multi-level and cross-domain security policies) will limit operational agility and effects while increasing operational risk to mission and force. Adversaries continue to gain the ability to make faster tactical and operational decisions; when the Air Force effectively integrates multi-domain effects through a resilient kill mesh/network (replacing a linear "kill chain"), capable of operating at the tactical edge, the joint force should realize a decision advantage over peer competitors. When fully leveraged, the continuum across domains of battle management and command and control (BMC2) capabilities and professionals provide the unparalleled perspective, insistent presence, and processing required to (1) manage information, (2) orient systems, (3) synchronize effects, and (4) speed decisions across the spectrum of conflict.

As distinct advanced combat capabilities continue to drive employment concepts, disconnects (intelligence, information, effects-based planning, briefing, execution, and debriefing (PBED)) within the air, space, and cyber domains increase ("All-domain" operations is distinct from "multi-domain" operations). Despite advances in automation, machine learning, and artificial intelligence, man-in-the-loop requirements still exist as a result of C2 policies, architecture, infrastructure, training, organization, and systems limitations.

To address current shortfalls and ensure continued multi-domain relevance and efficacy, the Air Force BMC2 enterprise, IA policies, and conditions-based authorities' paradigm must evolve. The three primary objectives of this vision are:

1. Multi-domain weapons systems must be scalable and be able to integrate and be employed at the edge and across multiple levels of security.
2. Capabilities must be survivable and operate with open architectures to allow for faster integration of technology providing multi-domain, joint/coalition interoperability.
3. Tailored and automated integration of INTEL, CYBER, and SPACE personnel and systems must present easily understood courses of action to operators and decision makers. This includes an assessment/feedback mechanism that allows branches and/or sequels.

Observe-Orient-Decide-Act Loop Architecture

Observe-Orient-Decide-Act (OODA) was introduced by Colonel John Boyd, USAF in the 1950's to help train fighter pilots and has since been extended to a wide array of strategic scenarios encompassing both individual decision making and the collective.

Observe: build a comprehensive picture of the situation with as much accuracy as possible.

Orient: find mismatches: errors in your previous judgement or in the judgement of others. Generally, bad news is the best kind provided that you catch it in time, as you can turn it to your advantage.

Decide: having gathered information and oriented ourselves, we must make an informed decision. The previous two steps should have generated a plethora of ideas, so this is the point where we choose the most relevant option.

Act: execute the decision and then **Observe** the results.

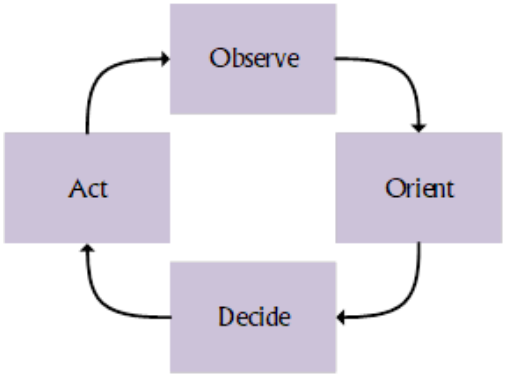

Figure 2 – Classic OODA Loop

Multiplex OODA Component View

To achieve a solution to the problems described on page 3 we must identify several components in each frame of reference. In the Observe frame we Validate raw data which is then Analyzed and Presented in the Orient frame. The enhanced data is then used in Decision Modeling in the Decide frame and that decision is Distributed and Acted upon in the Act frame.

Multiplex will focus on Validate and Analyze components.

Observe—Validate

The Validation component applies metadata to raw inputs for further processing and can include transformations and combining of data.

Orient—Analyze

The Analysis component takes the output of Observation and applies Artificial Intelligence/Machine Learning to produce details for enhanced decision making.

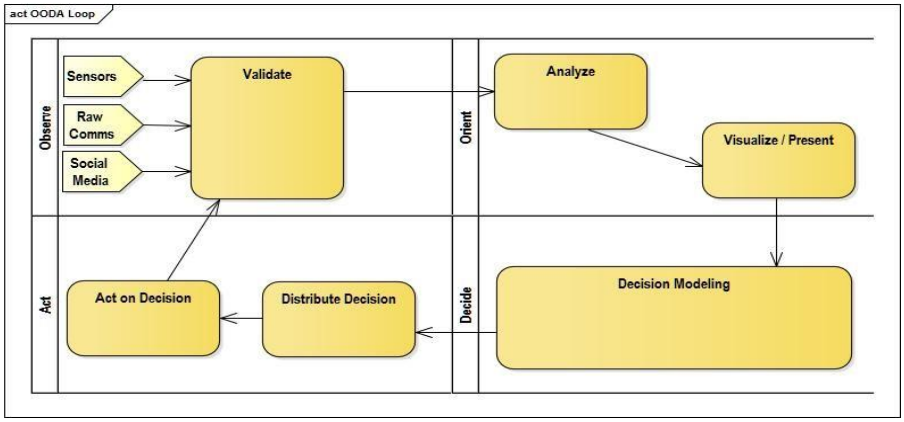

Figure 3 – MDO OODA Loop

An Introduction to Fog Computing

The OpenFog Consortium was founded by Microsoft, Intel, Dell, Cisco, and others in 2015 and defines Fog Computing as “A horizontal, system-level architecture that distributes computing, storage, control, and networking functions closer to the users along a cloud-to-thing continuum”.

Fog computing is an extension of the traditional cloud-based computing model where implementations of the architecture can reside in multiple layers of a network’s topology. However, all the benefits of cloud should be preserved with these extensions to fog, including containerization, virtualization, orchestration, manageability, and efficiency. In many cases, fog computing works with cloud.

In the notional diagram below, Fog nodes exist on vehicles, drones, aircraft--even the warfighter, to communicate and control IoT sensors. These nodes would talk to each other and role up to the base FedRAMP cloud. This topology is ideal for resiliency in validation and analysis in D-DIL environments and refresh data models when connectivity is restored.

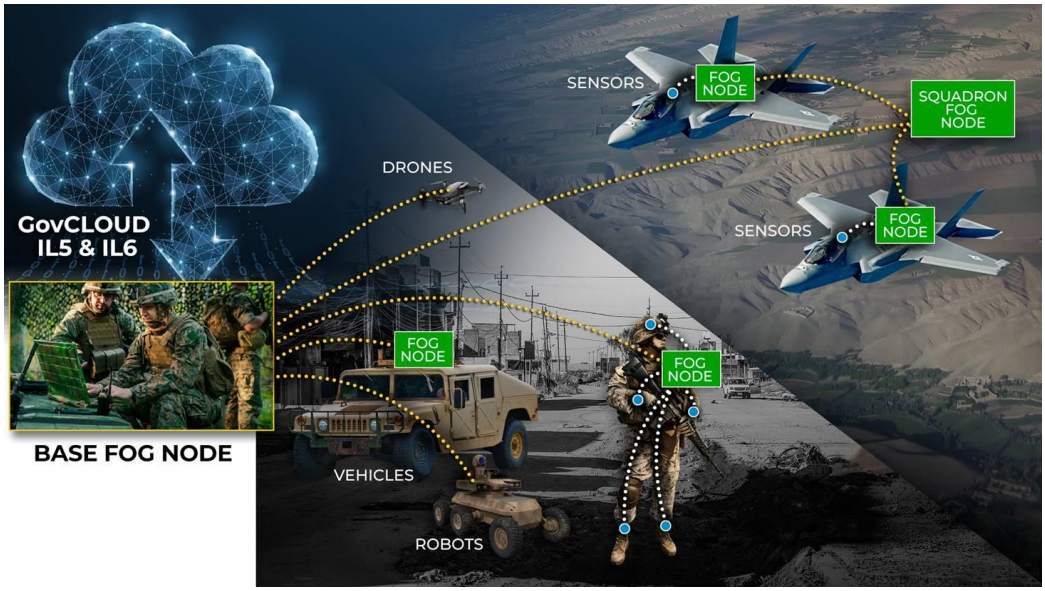

Figure 4 – Notional Battlefield Fog

The solution we will describe in the subsequent pages will present a Data Architecture that leverages the OpenFog concept to provide Capture, Validation, Analysis and Multi-Level Secure Distribution of arbitrary Data while providing and an example workflow that leverages combining data from multiple sensors and feed that result to an Artificial Intelligent model while tagging the various data packages with classifications and distributing those. A notional fog architecture is shown below.

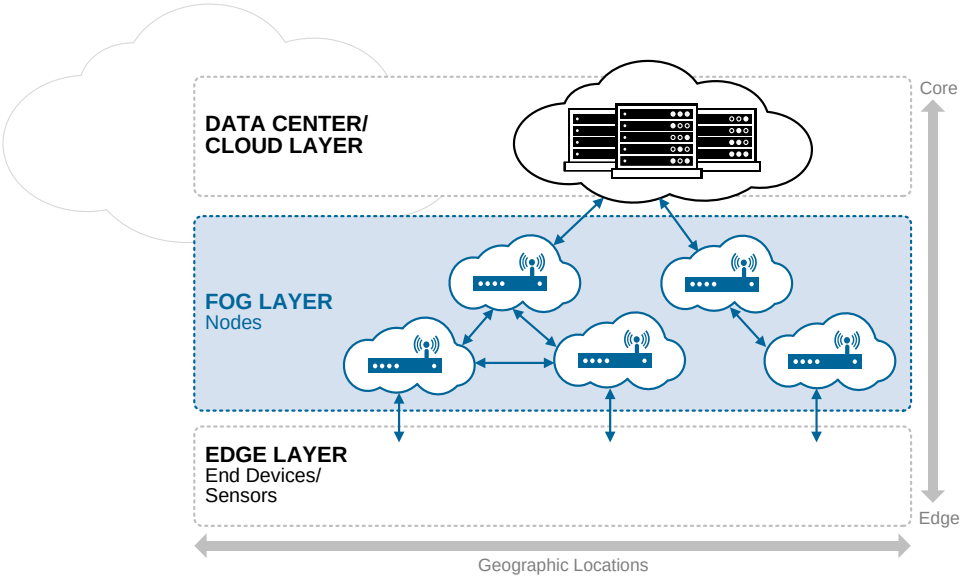

Figure 5 – Fog role in the Tactical Information Stream

The Bureaucratic Data Silos

The vision of a DoD unified data fabric that has broken down the data silos within DoD components, disciplines, and layers of bureaucracy is obtainable. The intelligence community, at the direction of the Director of National Intelligence, has been working on such a data fabric using cloud technologies as the foundation. We will frame the problem that data is currently stored in thousands of NetApp, Dell, EMC, Avamar, 3Par, Oracle, and many other vendors' storage appliances within their data centers. The majority of data is unused or has not been accessed for years. Some of this is due to regulatory requirements from the National Archives, HIPAA, and other US Codes. Making matters worse, most DoD organizations use SharePoint for a document library with SQL back-ends, which makes the least valuable data the most expensive and hardest to actually use without writing complicated structured query language (SQL) code based on custom database schemas. The most valuable data is the data that is part of the OODA loop that is relative to the organizational mission today; which in the DoD is to win or prevent wars. We must always be mindful of Conway's Law, which at its premise describes that our applications and infrastructure (including data) are representative of the way we communicate in an organization. The current state of DoD data is therefore a massive bureaucratic sprawl that requires many manual processes built to make it very difficult to share data.

The data silos and bureaucracy can be broken down by moving the data into a unified DoD data lake and leveraged to discover new insights by applying machine-learning models. This data should be auto-tiered by cloud providers based on usage and value to the organization. The Observe and Orient part of the OODA loop can then use this historical data to provide additional context to new data coming from current operational missions. This new data that is to be stored in a cloud service provider must be pushed through data pipelines to extract the maximum value. These pipelines should include capabilities such as Object Recognition, Optical Character Recognition, Natural Language Translation, Sentiment Analysis, and other organization specific machine learning models developed. The data pipelines should be applied closest to the source of the data allowing faster OODA loop to occur. This enables commanders and tactical consumers of the intelligence extracted from the data to seize the initiative and achieve faster insight. This means, not only having a faster OODA loop then the opponent force, but achieving overmatch by presenting additional dilemmas by discovering more opportunities to exploit. The closest source of valuable data is the data at the tactical level and cuts across all layers of MDO.

Fog Computing and Data Locality

The use of Fog Computing allows these data pipelines to be built across the tactical level and ensures that data is enriched, verified, and leveraged for immediate intelligence and targeting. Data locality and data gravity are important considerations even in a world with gigabit LEO satellite systems and 5G. Data locality ensures that the data is stored closest to the location where it is processed and consumed. Data gravity is a theory that states that applications and likewise machine learning should be closest to where the preponderance of data is physically located. This is because the DoD network (e.g. DoDIN, MPE) or internet may not always be available as many have learned trying to access SharePoint over a SATCOM link primarily due to latency. Even when a reliable network is available and latency is very low (< 5ms), the use case for Fog Computing remains to allow for efficient use of bandwidth and survival of capability when Electronic Warfare (EW) tactics are used. The main difference between Fog Computing and Edge Computing is that Fog Computing is distributed mesh network with tiers, while Edge Computing is not and relies completely on the wide area network to share data and intelligence. When the wide area network is available Fog Computing ensures that it is efficiently used to move data into a cloud service provider were further long-term analysis can be conducted.

SmartFog Platform & Microservices

Technica's Independent Research and Development (IR&D) division developed the SmartFog prototype platform to bring functions (compute, storage, networking, AI-acceleration, analytics, and management control) closer to the edge—where the IoT devices reside.

This allows IoT events to be processed in near real-time. Importantly, SmartFog allows for data localization, i.e. data can be processed near the edge. This offloads some of the analytics burden from the cloud or core on-premise datacenters. Faster results are obtained, and with less security risk, than transmitting all data to central servers for processing.

SmartFog Microservices Catalog

SmartFog Microservices provide discrete functions and can be viewed as analogous to apps on smartphones. However, unlike most smartphone apps, SmartFog Microservices can pass messages between themselves to create composite microservices. For example, the Anomaly Detection Microservice (described below) can communicate with the Complex Event Processor (CEP) Microservice to trigger alarms or alerts. Microservices can be dynamically updated, like upgrading apps on smartphones.

While SmartFog is a powerful platform, the platform is only as powerful as the jobs it can perform. Thus, the innovations involved in creating the platform, e.g., enabling Docker and Singularity images on the NVIDIA TX2 that can be accelerated using graphics processing units (GPUs), are matched with microservices, to provide Message Queuing Telemetry Transport (MQTT), message brokering, anomaly detection, and federated learning.

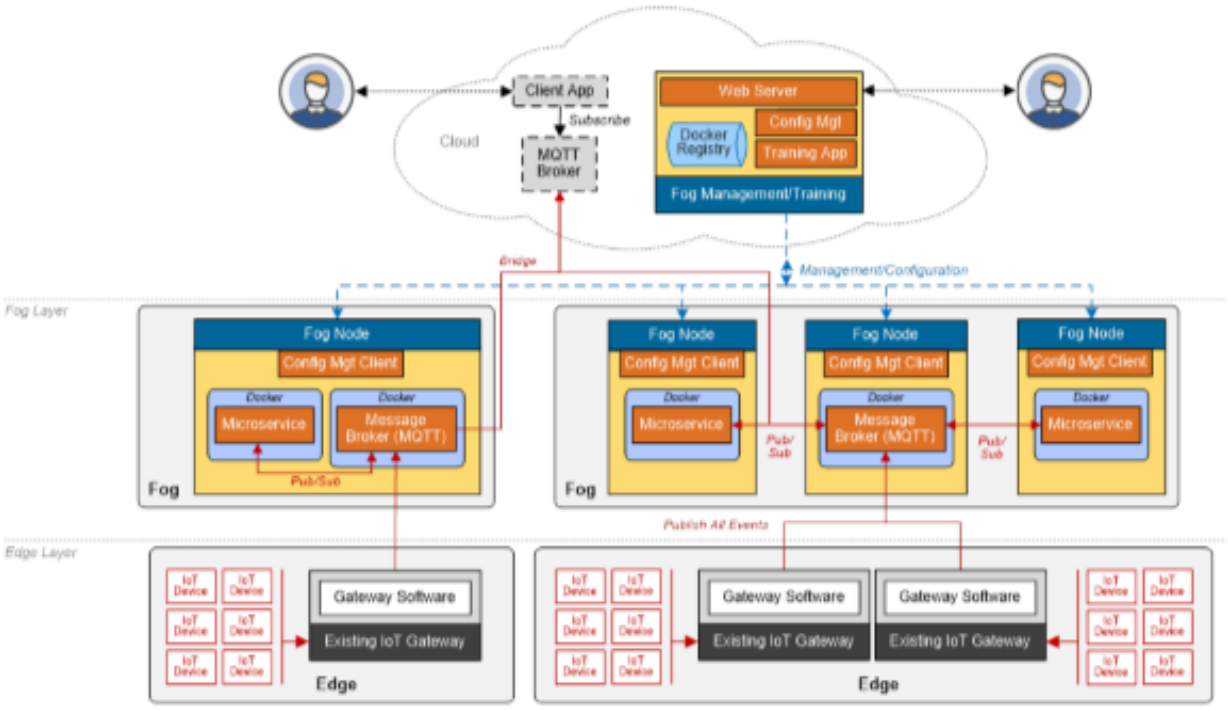

Figure 6 – Representative SmartFog Architecture

While a microservice can be created for nearly everything— just like smartphone apps—Technica sees the greatest value in providing microservices that take advantage of hardware acceleration and offer AI capabilities with Deep Learning algorithms.

Anomaly Detection

The Anomaly Detection Microservice (ADM) uses a specific neural network architecture—an autoencoder—to compress and decompress data as shown in the figure to the right.

The orange neural layers in the figure reduce the input into a compressed feature vector. The green neural layers attempt to expand the feature vector and recreate it. Since the network is trained to reproduce common data easier, data with more decompression errors is identified as anomalous. This neural network construction can be used to detect outliers in any signal that can be represented mathematically, e.g., network traffic packet capture data, radio frequency data, Humvee sensor data (engine temperature, tire pressure), etc.

There are two major steps to implementing a Deep Learning based algorithm like Anomaly Detection.

- Training—the process in which the Neural Network learns from the data to expect "normal" conditions.
- Inference—once the training is complete, the model is deployed. Real-time data is then processed by the model.

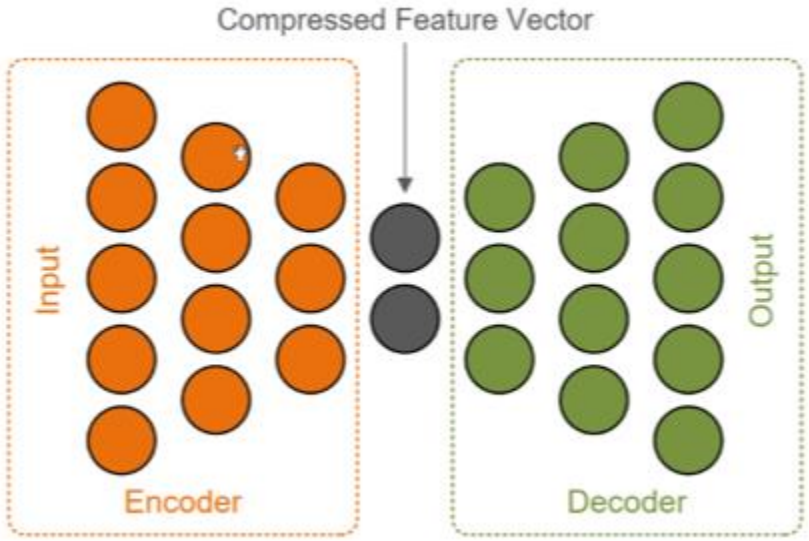

Figure 7 – Autoencoder

The inference process for the Anomaly Detection Microservice involves calculating an anomaly score. The parameters of this score are configurable. In other words, the threshold at which an Anomaly is detected is configured by the user.

The training and inference workflow is basic to all Deep Learning algorithms. This process typically is continuous, such that the AI models steadily improve over time. At a minimum, the training is usually done in the cloud. Often, inference is also done in the cloud. However, one the central motivating factors for the creation of SmartFog was the desire to deliver AI capabilities to the tactical edge. Given D-DIL constraints, a continuously improving AI model becomes problematic because a cloud connection is not always available.

For these reasons, Technica developed a series of SmartFog Microservices to provide Federated Learning.

Federated Learning

Using the traditional approach, training a neural network requires having a single copy of the model and all the training data in one place. In many real-world scenarios, data is gathered across an array of sensors. In those scenarios, all sensor data would have to be sent to a central server for training and the resulting network weights would have to be distributed back to the sensors. However, these sensors often have limited bandwidth and intermittent connections to the central server. Federated learning allows the model to be trained on each edge device and is based on the data parallelism model.

In the data parallelism model, multiple copies of the neural network are created. The training data is split between the copies of the network, such that each copy is trained on an independent section of data. Once all copies of the model are trained, the resulting weights are aggregated at a central repository. This is usually accomplished by averaging the weights of the independently trained models. It is easy to see how this translates to edge computing, where each edge device trains a copy of the neural network using the data that it observes. The new network weights are sent to the central server for aggregation, and the resulting model is then distributed to the edge devices. In this way, the model at each edge device will have learned from the data gathered from all devices without having to transmit the full training to the central server. This greatly reduces bandwidth requirements and can occur when a connection to the central server is available.

Technica's anomaly detection on the edge model is comprised of two elements, as seen in the figure below. One is a distributed set of devices each deployed with a copy of the autoencoder and the other is a central server for aggregating and distributing changes to the trained autoencoder network weights. Multiple rounds of training and aggregation may occur in order to produce better results. The edge devices independently gather observations and determine an anomaly score for each.

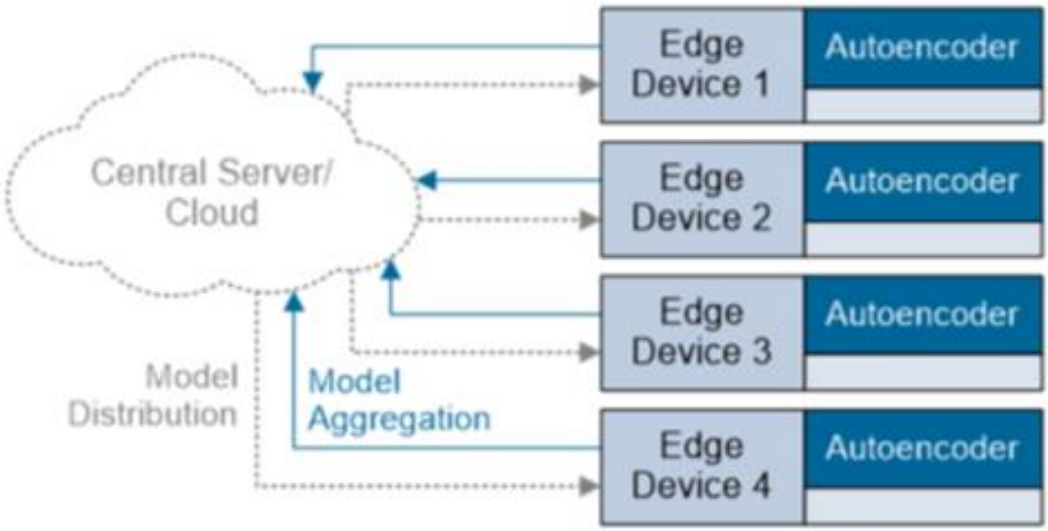

Figure 8 – Federated Learning representation

High Level Data Architecture

A key requirement of the AFWERX MDO Challenge is that the solutions be data source agnostic. We have identified that NIPR, SIPR, JWIC/NSANet, some SAP/SAR nets and Coalition Networks will be likely data sources.

Data is collected in real-time, validated, and analyzed at both the edge and in the cloud. The edge consists of dedicated servers that sit close to the action, where communications to individuals and sensors is less likely to be disrupted while also conserving uplink bandwidth. The cloud will offer significantly more powerful computing resources and more advanced analysis; however, it is typically only intermittently (at best) available in the field.

As data is collected in real time at the edge it is then validated at the edge producing artifacts such as additional organization, filtering and security attributes in a data set which is then queued for transfer to the cloud where additional validation, such as translating the data into common data schemas, or comparative sensor integrity, in order to validate signals from other edges regarding the same event against each other and to detect enemy tampering with signals. Analysis then occurs at both the edge and in the cloud, applying AI and ML to aid in decision making, or to suggest actions, from units in the field to senior leadership.

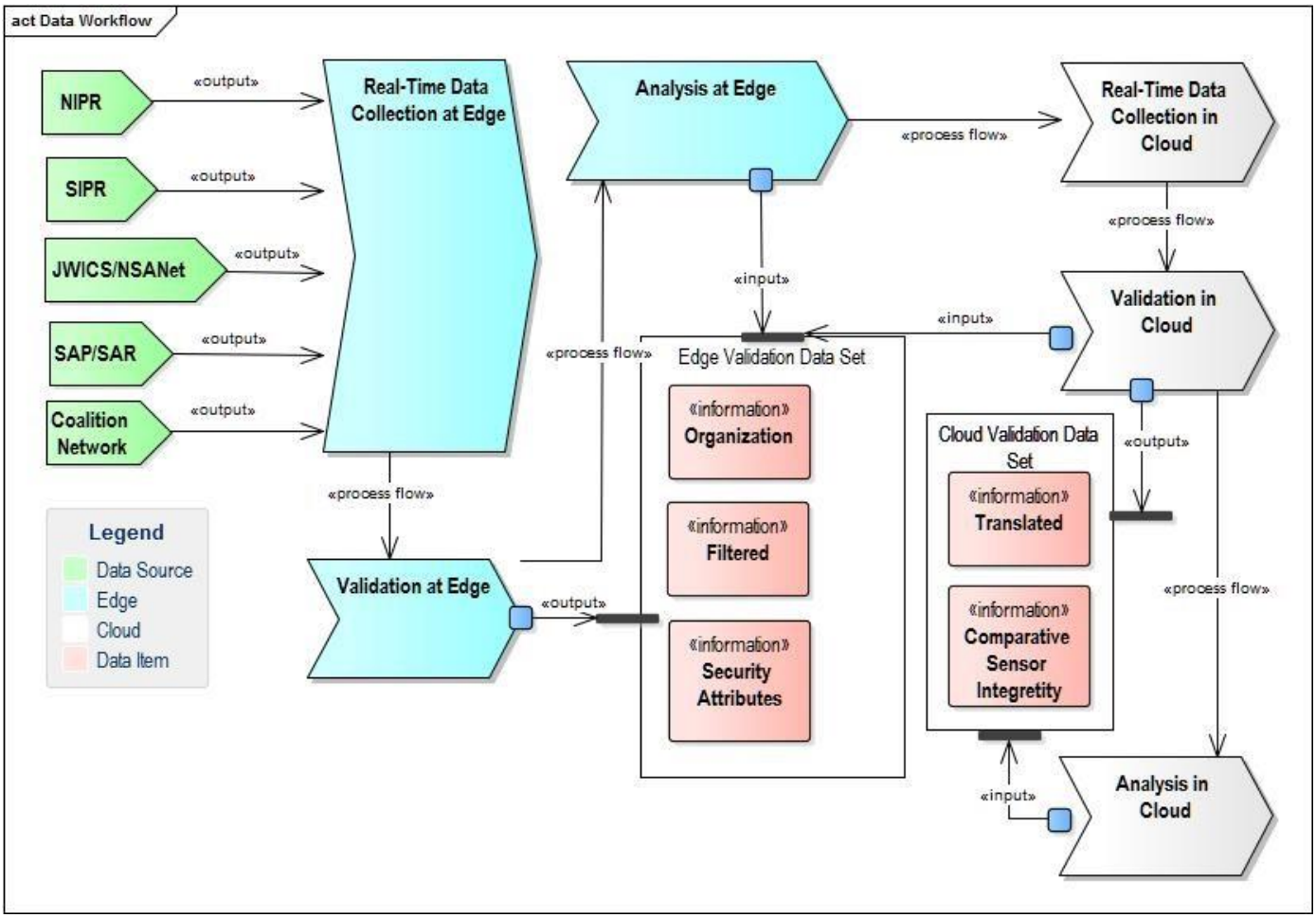

Figure 9 – Hi-Level Data Architecture

Streaming Architecture

Fog nodes, running the SmartFog Platform, will host Kafka clusters. Apache Kafka is a stream-processor that provides a unified, high-throughput, low-latency platform for handling real-time data feeds. Kafka will be connected to various sensors in the fog layer producing streams whose broadcast is captured by a validation component that writes validated information back to Kafka. Data will then be classified by the Basil distributed policy system and pushed to the cloud which implements a similar workflow with additional analysis, including AI/ML processing.

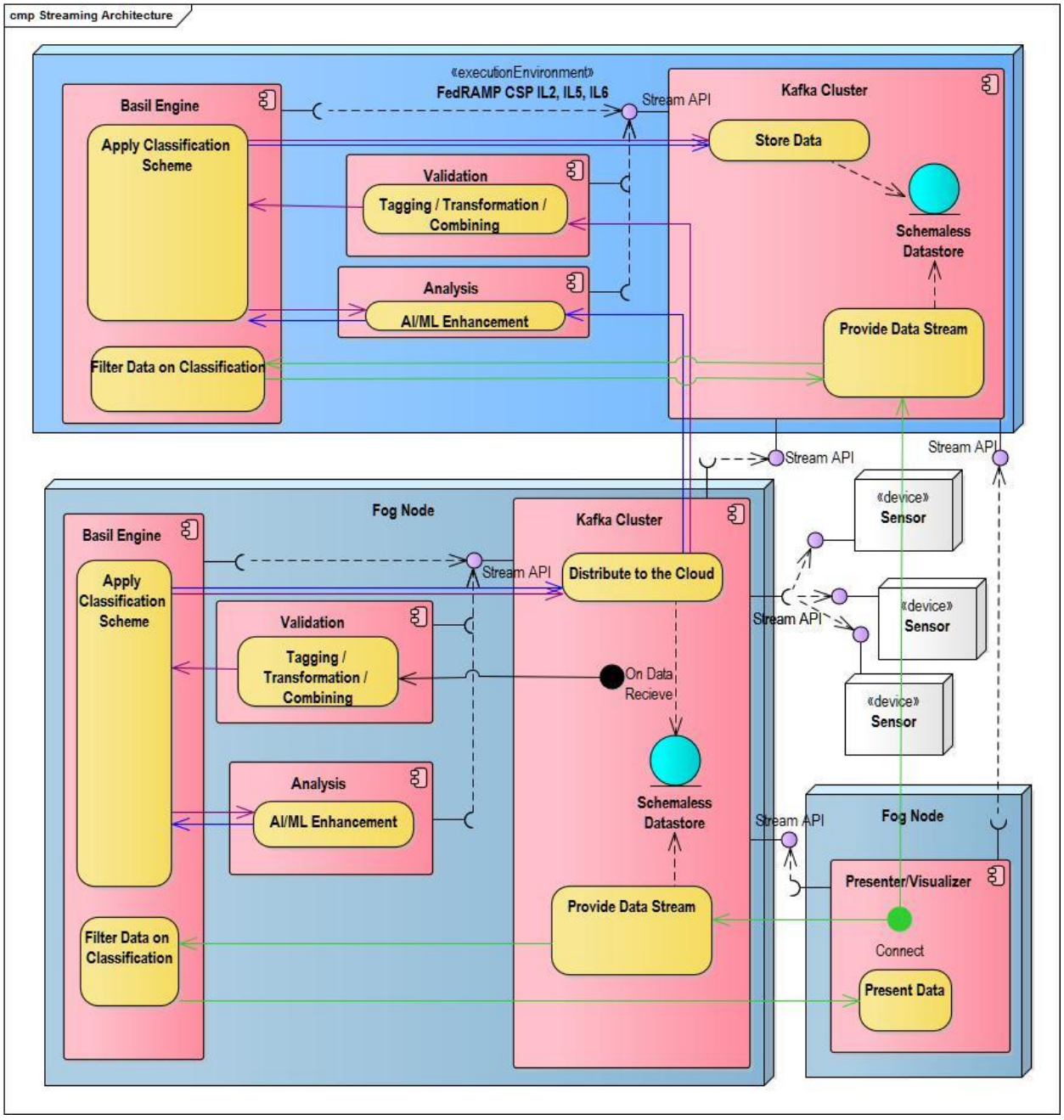

Figure 10 –Streaming Architecture

Multi-Level Security, Encryption, and Information Classification

MDO involves multiple technical systems, which have different, detailed security requirements. Across all components of the complete, integrated MDO system, established security mechanisms and best practices must be implemented, for example, encryption of data in flight and at rest. However, the scope of this section is not to describe a general security architecture that implements well known security mechanisms and best practices. The proposed solution, while it shall include well known security mechanisms and best practices, is based on the Basil policy-as-code platform.

Basil Policy-as-code Platform

Basil is an advanced, high performance policy-as-code framework for distributed security policy enforcement that combines next-generation NIST attribute-based access control (ABAC) with event-driven and stateful security policy conditions. Event driven conditions refer to information that triggers security policy decisions communicated to the policy system from an external system based on a change in state in the external system. Stateful conditions refer to information that the policy system must request from external systems to determine if their current state meets policy requirements. Basil policies are referred to as “Smart Policies” because, unlike static access control rules, Basil policies adapt automatically under changing conditions, based on events or external state information.

Basil addresses many aspects of MDO security because it serves both as an integrated component, e.g., for information classification and technical operations security, and also as an overlay able to enforce security policies over otherwise disparate systems. Basil provides single-pane-of-glass visibility into policies across the complete, integrated MDO system.

Zero Trust Security and Operations

MDO involves the operation of and privileged access to systems, environments, applications, and sensitive data, as well as to application programming interfaces (APIs), code, and digital secrets, such as API and encryption keys. MDO requires multiple security levels that must operate across all levels of US and allied militaries. Zero Trust is a security model based on the principle of maintaining strict access controls for every interaction within a system and of not trusting any system or individual by default, even those already inside the network perimeter. Basil is a zero-trust system where every action by every actor, whether human or machine, is subject to policy evaluation.

Multi-level Access Control

As a policy system, Basil is not a conventional identity and access management (IAM) system. Basil assumes the use of external standards-based authentication (AuthN) and authorization (AuthZ) using the OAuth and JavaScript Object Notation (JSON) Web Tokens (JWT) standards together with OpenID Connect (OIDC). Using this model, each MDO coalition partner can act as their own authoritative identity provider (IdP) without any single party having control over an aggregate set of credentials.

Internally, Basil creates a unique asymmetric private key for each actor and every action within Basil is digitally signed. Basil policies can be used to define “Synthetic Roles” using any collection of attributes associated with actors, and to emulate role-based access control (RBAC). Synthetic Roles can be used to create a 1:1 mapping of RBAC if desired. Additionally, Basil supports fine grained differences between actors based on (1) arbitrary combinations of attributes that may or may not correspond to RBAC roles; and (2) the state of any attribute referring to an external system. For example, the length of time since the last communication with a system could be used to change the access level of an actor or actors.

Policy Integrity Across Systems

In most systems, particularly in systems of systems, such as MDO, there is typically no provable link between security and operational policies and the implementation of policies in constituent systems. Auditing often involves manual data collection and analysis and can be time consuming. Specifically, auditing is anti-agile and can delay adaptation of a system to operate in a wide variety of contexts with minimal effort.

Basil creates a chain of integrity for policy enforcement across systems. In order to facilitate distributed replication of policy information in a strongly consistent manner, Basil includes plugin drivers for several different database systems. The default plugin is a blockchain based datastore, which is a private, permissioned blockchain that utilizes a proof of authority (PoA) consensus algorithm based on Parity. The blockchain enables the policy system to act as a distributed state machine where nodes within the policy system are cryptographically guaranteed to observe a single, consistent set of policies.

Unified, Immutable Audit Logging

A problem related to policy integrity is unified audit logging. In most systems, particularly in systems of systems, security and operations auditing is highly fragmented because different components generate different log data and use different security and operational mechanisms. Basil enforces policies over access and serves as an execution context for otherwise disparate tools and APIs. As a result, Basil maintains a unified audit log and ensures accountability for all actions, whether by humans or machines.

Using a blockchain, logging of policy changes, approvals, and other actions is immutable, which means that the logged information cannot be tampered with or erased. Combined with non-repudiation, based on the use of digital signatures, blockchain based logging constitutes digital forensic evidence ensuring the security of log data, as well as the accountability of all actors, whether human or machine.

In addition to Basil's own logs, it is possible to record the hash values of external logs on the blockchain so that the integrity of the logs can be proved. When there is an output of interest, the log output can be optionally hashed with a cryptographically secure hash function, and the hash can be included in the log database.

Since Basil is an ABAC policy system, policies can be used to provide selective views of audit data. For example, MDO coalition partners can be optionally permitted to view audit data related to their own forces, or other select forces.

Human Readable Policy Language

The Basil policy language (BPL) is a human-readable format similar to an infrastructure-as-code (IaC) domain specific language (DSL). BPL is a variant of the opensource HashiCorp Config Language (HCL), which is a popular opensource language used by HashiCorp Terraform. Terraform is an IaC software tool that allows users to define and provision infrastructure as a service (IaaS) in a simple, declarative way. BPL is well suited to MDO use cases because it is relatively easy to understand. Additionally, web-based forms can be created to manage Basil policies, rather than relying entirely on BPL source code.

Fault Tolerance and Resilience under D-DIL Conditions

In the MDO system, D-DIL communications are expected. Additionally, components may fail, or may be compromised, and there may be imperfect information on whether a component has failed or is compromised. The technical term for these conditions is Byzantine failure. A consensus algorithm represents a strategy to avoid catastrophic system failure under Byzantine failure conditions. Basil is a blockchain-based, distributed system that operates as a set of nodes linked by Internet Protocol (IP) networks. Since it is Byzantine fault tolerant, Basil nodes can be added or removed from the network and communication can be disrupted or restored without compromising the overall integrity or operation of the system. If Basil nodes are compromised, the consensus algorithm and cryptographic mechanisms guarantee the integrity and consistency of policies.

Use of Basil for MDO

Basil is a general purpose policy management and enforcement system that can be used to implement security controls over multiple, normally disparate areas, such as application logic and hardware configuration. For MDO, six (6) Basil use cases have so far been identified:

- Application Security
- Automation Security
- Development and Technical Operations Security
- Hardware Configuration Security
- Policy-based Information Classification
- Multi-level Encryption

Application Security

MDO applications can obtain security policy verdicts through the Basil representational state transfer (REST) API or Basil can be used as a proxy for API calls. Additionally, Basil can consume webhooks and, through its plugin system, can be integrated with message queuing systems or event stream-processing software platforms, such as Apache Kafka, which will allow Basil to be informed of changing conditions and to take actions as a result pursuant to defined policies.

Automation Security

Basil will be used to validate machine-to-machine interactions as data flows through MDO systems to the cloud. Within Basil, human and machine actors are equivalent for the purpose of policy evaluation. Webhooks are a common form of interaction natively supported in the Basil AuthN/AuthZ model, allowing Basil to enforce rich policy descriptions on machine-to-machine interactions. This may include selective forwarding of webhooks based on external or internal information. An example use case of this capability is access control logic when integrating automated build and test (continuous integration) systems with source control systems, which is an often overlooked attack surface.

Development and Technical Operations Security

Basil addresses the ongoing development and operation of the complete, integrated MDO system. The need to develop software and to make software changes faster has led to high levels of automation, i.e., continuous integration and continuous delivery (CI/CD), and to the integration of development and operations (DevOps), particularly for software as a service (SaaS). However, neither automation in general, CI/CD in particular, nor DevOps make software or systems more secure or prevent erroneous or malicious actions by insiders.

Basil will be used to institute and enforce new security controls over MDO DevOps activities such as managing automated CI/CD pipelines, cloud infrastructure, orchestration systems, e.g., Kubernetes, and the related use of digital secrets, such as API keys, certificates, and encryption keys.

Hardware Configuration Security

Hardware devices, such as network routers, on-board computers, and other smart equipment, often support dynamic configuration through RESTful APIs, e.g., IoT devices, routers, and network switches. Misconfiguration of hardware devices can result in large scale exploits or system failures, potentially on a global scale. Basil will be used to enforce strict control over hardware configuration changes and to establish failsafe policies designed to prevent catastrophic failures. If a hardware device does not expose a RESTful API, Basil can leverage Secure Shell (SSH) or another mechanism.

Policy-based Information Classification

Security related attributes will be associated with data as early as possible so that security attributes can be used to control access using Basil policies at the tactical level. As information is aggregated, combined, or, later, processed in the cloud, the security attributes in each data set can be compared to the attributes of human or machine actors attempting to access the data.

In Basil, a set of information classification policies will be defined based on data attributes. Smart Policy evaluation may require Basil to obtain information from other systems in order to complete a policy evaluation. As data streams in from the edge from different data sources, it will be packaged in standard formats, such as JSON documents, and will have associated metadata attributes. As a part of the process of filtering and of assigning attributes to data, API calls will be made to Basil to request a classification policy verdict. Policy evaluations do not require blockchain consensus, thus API calls to the Basil Server are similar to other scalable microservices.

Multi-level Data Encryption

Using Basil, the MDO system can encrypt data using different keys for different coalition partners and classification levels corresponding to the attributes associated with the data and with actors in the Basil system. Basil policies can use combinations of attributes to evaluate requests for unencrypted data access. In this context, it is possible for coalition partners to, optionally, provide their own encryption keys when they are on-boarded into the MDO system.

Since Basil can execute arbitrary code as the result of a policy evaluation, it can be used to encrypt and to decrypt data on demand. The advantage of using Basil to manage data encryption and decryption is that Basil isolates encryption keys from actors. Basil can be integrated with key management systems or credential vaults and can be used to execute code on behalf of any actors pursuant to a policy evaluation. Actors do not require access to secret key material. Additionally, within Basil, digital secrets are a special data type, represented in all cases except code execution using ephemeral tokens which can be safely reflected in logs and which cannot be used to compromise the Basil system.