2. SABSA Framework Overview

The Sherwood Applied Business Security Architecture (SABSA) is a proven methodology for developing business-driven, risk-managed security architectures at both enterprise and solutions levels. When applied to API authorization using Ping Authorize, SABSA provides a comprehensive framework that ensures security measures are aligned with business objectives and effectively manage risks.

2.1 Introduction to SABSA

SABSA is a layered model that addresses security architecture from multiple perspectives. It provides a systematic approach to security design, implementation, and management, ensuring that all aspects of an organization's security are considered and aligned.

2.1.1 Key Principles of SABSA

1. **Business-Driven**: All security decisions are based on identified business needs and drivers.
2. **Risk-Based**: Security measures are designed to address specific, identified risks.
3. **Enterprise-Wide**: The framework considers security across the entire organization, not just in isolated silos.
4. **Lifecycle Approach**: SABSA covers the entire lifecycle of security architecture, from conception to retirement.
5. **Scalable and Adaptable**: The framework can be applied to organizations of any size and can adapt to changing business needs.

2.1.2 Benefits of Using SABSA for API Authorization

- Ensures API security strategies align with overall business objectives
- Provides a structured approach to identifying and addressing API-specific risks
- Facilitates communication between technical teams and business stakeholders
- Enables the development of a comprehensive, layered security approach for APIs
- Supports the creation of measurable security outcomes

2.2 SABSA Layers

SABSA defines six layers of security architecture, each addressing a different perspective of the security challenge. When applied to API authorization using Ping Authorize, these layers

help ensure a comprehensive and well-structured approach.

2.2.1 Contextual Architecture (Business View)

The Contextual Layer focuses on understanding the business context and requirements for API security.

Key Questions:

- What are the business drivers for securing our APIs?
- Who are the stakeholders involved in API usage and security?
- What are the potential impacts of API security breaches on our business?

Application to Ping Authorize:
At this layer, we identify the high-level requirements for API authorization. For example:

- Need for fine-grained access control to protect sensitive data exposed via APIs
- Compliance requirements (e.g., GDPR, PCI-DSS) that impact API access
- Business partnerships that require secure API integrations

Example Outcome:
A documented set of business requirements for API authorization, such as:
"Our API authorization system must support dynamic, context-aware access decisions to meet the varying security needs of our diverse API portfolio while ensuring compliance with data protection regulations."

2.2.2 Conceptual Architecture (Architect's View)

The Conceptual Layer defines the fundamental concepts and principles that will guide the API authorization strategy.

Key Questions:

- What are the core security principles we need to apply to our APIs?
- How do we conceptualize the relationship between API consumers, resources, and access rights?
- What high-level security services are needed to protect our APIs?

Application to Ping Authorize:
Here, we define the conceptual framework for API authorization. This might include:

- Adopting the principle of least privilege for API access
- Conceptualizing a zero-trust model for API interactions

- Defining the basic elements of our authorization model (e.g., subjects, resources, actions, conditions)

Example Outcome:
A conceptual model for API authorization, such as:
"Our API authorization will be based on an Attribute-Based Access Control (ABAC) model, allowing for dynamic, context-aware decisions. It will incorporate elements of zero-trust, requiring continuous validation of every API request."

2.2.3 Logical Architecture (Designer's View)

The Logical Layer translates the conceptual model into logical structures and processes, independent of specific technologies.

Key Questions:

- What logical components are needed to implement our API authorization strategy?
- How will authorization decisions be made and enforced?
- What information flows are required to support API authorization?

Application to Ping Authorize:
At this layer, we design the logical components of our authorization system:

- Defining the structure of authorization policies
- Designing the logical flow of an authorization decision
- Specifying the attributes needed for access decisions

Example Outcome:
A logical architecture diagram showing the components of the API authorization system:
[Insert diagram showing Policy Administration Point (PAP), Policy Decision Point (PDP), Policy Enforcement Point (PEP), and Policy Information Point (PIP), and their interactions]

2.2.4 Physical Architecture (Builder's View)

The Physical Layer specifies the actual technologies and products that will implement the logical architecture.

Key Questions:

- How will Ping Authorize be deployed to support our authorization needs?
- What other systems need to integrate with Ping Authorize?
- How will we ensure the performance and scalability of our authorization system?

Application to Ping Authorize:
Here, we make specific technology choices and design the physical implementation:

- Specifying the deployment architecture for Ping Authorize
- Designing integrations with API gateways, identity providers, and other systems
- Determining hardware and network requirements

Example Outcome:
A detailed technical architecture document, including:

- Deployment diagrams for Ping Authorize components
- Integration specifications for connecting Ping Authorize with existing API infrastructure
- Performance and scalability considerations, such as caching strategies and load balancing

2.2.5 Component Architecture (Tradesman's View)

The Component Layer focuses on the detailed configuration and implementation of individual system components.

Key Questions:

- How should each component of Ping Authorize be configured?
- What specific policies need to be implemented?
- How will attributes be defined and managed within the system?

Application to Ping Authorize:
At this layer, we dive into the specifics of configuring and customizing Ping Authorize:

- Defining detailed authorization policies using Ping Authorize's policy language
- Configuring attribute sources and mappings
- Setting up logging and monitoring

Example Outcome:
Detailed configuration specifications, such as:

- Sample policy definitions in Ping Authorize's format
- Attribute mapping tables
- Configuration scripts or templates

2.2.6 Operational Architecture (Facilities Manager's View)

The Operational Layer addresses the ongoing management and operation of the API authorization system.

Key Questions:

- How will we monitor the effectiveness of our authorization policies?
- What processes are needed for updating and maintaining policies?
- How will we respond to authorization-related incidents?

Application to Ping Authorize:
Here, we define the operational processes and procedures:

- Establishing monitoring and alerting for authorization decisions
- Defining processes for policy updates and version control
- Creating incident response plans for authorization-related issues

Example Outcome:
An operations manual that includes:

- Monitoring dashboards and KPIs for API authorization
- Policy management workflows
- Incident response playbooks for common authorization scenarios

2.3 SABSA Matrix

The SABSA Matrix is a powerful tool that helps ensure all aspects of security architecture are addressed across all layers. It consists of six columns (Assets, Motivation, Process, People, Location, Time) applied to each of the six layers.

2.3.1 Applying the SABSA Matrix to API Authorization

Let's explore how the SABSA Matrix can be applied to API authorization using Ping Authorize:

Layer	Assets	Motivation	Process	People	Location
Contextual	What API resources are we protecting?	Why do we need to secure our APIs?	What business processes involve API usage?	Who are the stakeholders in API security?	Where are our APIs accessed from?
Conceptual	What is our conceptual model for API resources?	What security goals are we trying to achieve?	What are the key processes in our authorization model?	What roles are involved in API authorization?	What is our conceptual view of API access locations?

Layer	Assets	Motivation	Process	People	Location
Logical	How do we logically represent API resources?	How do we logically structure our security policies?	What are the logical steps in making an authorization decision?	How do we logically represent users and roles?	How do we logically define location in our policies?
Physical	How are API resources represented in Ping Authorize?	How are security policies implemented in Ping Authorize?	How does Ping Authorize process authorization requests?	How are user attributes stored and retrieved?	How does Ping Authorize determine the location of API requests?
Component	How are individual API endpoints configured?	How are specific policy rules defined?	How are policy decision processes configured?	How are user directories integrated?	How are geolocation services configured?
Operational	How are API resources monitored and managed?	How is policy effectiveness measured?	How are authorization processes monitored?	How are user access rights reviewed and updated?	How is location-based access monitored?

This matrix helps ensure that all aspects of API authorization are considered at each layer of the architecture.

2.4 SABSA Lifecycle

The SABSA Lifecycle provides a continuous process for developing, implementing, and managing security architecture. It consists of several phases that align closely with the development and operation of API authorization systems.

2.4.1 Strategy and Planning

In this phase, the overall approach to API authorization is defined, aligning with business objectives and risk management strategies.

Key Activities:

- Conduct a business impact analysis for API security
- Develop a high-level API security strategy

- Define key performance indicators (KPIs) for API authorization

Application to Ping Authorize:

- Assess how Ping Authorize aligns with the organization's API strategy
- Identify key features of Ping Authorize that address specific business needs
- Plan the phased implementation of Ping Authorize across the API portfolio

2.4.2 Design

The Design phase involves creating the detailed architecture and policies for API authorization.

Key Activities:

- Develop the logical and physical architecture for API authorization
- Design authorization policies based on business requirements
- Create detailed integration designs for Ping Authorize

Application to Ping Authorize:

- Design the policy structure using Ping Authorize's policy language
- Create attribute mapping designs for integrating with existing data sources
- Develop integration designs for API gateways and identity providers

2.4.3 Implementation

This phase involves the actual deployment and configuration of the API authorization system.

Key Activities:

- Deploy Ping Authorize components according to the physical architecture
- Implement and test authorization policies
- Integrate Ping Authorize with other systems (e.g., API gateways, monitoring tools)

Application to Ping Authorize:

- Install and configure Ping Authorize servers
- Implement authorization policies in Ping Authorize's policy administration interface
- Set up integrations with API gateways for policy enforcement

2.4.4 Operations and Management

The Operations phase covers the day-to-day running and maintenance of the API authorization system.

Key Activities:

- Monitor authorization decisions and system performance
- Manage and update policies as needed
- Handle authorization-related incidents and issues

Application to Ping Authorize:

- Set up monitoring dashboards for Ping Authorize
- Establish processes for policy updates and versioning
- Develop and maintain runbooks for common operational tasks

2.4.5 Reviews and Audits

Regular reviews ensure that the API authorization system remains effective and aligned with business needs.

Key Activities:

- Conduct periodic reviews of authorization policies
- Perform security audits of the API authorization system
- Assess the alignment of the authorization system with business objectives

Application to Ping Authorize:

- Review Ping Authorize logs and reports to identify potential policy improvements
- Conduct audits of Ping Authorize configurations and integrations
- Assess the effectiveness of Ping Authorize in meeting business and security objectives

2.4.6 Continuous Improvement

This ongoing phase focuses on evolving and enhancing the API authorization system.

Key Activities:

- Gather feedback from stakeholders on the authorization system
- Identify areas for improvement in policies and processes
- Implement enhancements to the authorization architecture

Application to Ping Authorize:

- Stay updated on new features and capabilities of Ping Authorize

- Implement policy optimizations based on operational data
- Explore advanced features of Ping Authorize to enhance authorization capabilities

2.5 Applying SABSA Principles to Ping Authorize Implementation

To effectively apply SABSA principles to a Ping Authorize implementation, consider the following guidelines:

2.5.1 Business Alignment

- Ensure that every aspect of the Ping Authorize implementation can be traced back to a specific business requirement or risk mitigation strategy.
- Use the SABSA matrix to map business drivers to specific Ping Authorize features and configurations.

2.5.2 Risk-Based Approach

- Conduct a thorough risk assessment of your API ecosystem.
- Use Ping Authorize's policy framework to implement controls that directly address identified risks.
- Prioritize the implementation of policies based on risk levels.

2.5.3 Layered Security

- Implement multiple layers of security using Ping Authorize in conjunction with other security controls (e.g., API gateways, threat protection systems).
- Use Ping Authorize's attribute-based access control to implement fine-grained authorization at different levels (API, resource, operation).

2.5.4 Measurable Outcomes

- Define clear, measurable objectives for your API authorization system.
- Use Ping Authorize's logging and reporting capabilities to gather data on policy effectiveness and performance.
- Regularly review metrics to ensure the authorization system is meeting its objectives.

2.5.5 Continuous Adaptation

- Leverage Ping Authorize's flexible policy framework to quickly adapt to changing business needs and emerging threats.
- Implement a regular review cycle for authorization policies, aligned with the SABSA lifecycle.

2.6 Case Study: Applying SABSA to API Authorization with Ping Authorize

To illustrate the application of SABSA principles to API authorization using Ping Authorize, let's consider a fictional case study of a financial services company, "SecureBank."

Background:

SecureBank is launching a new Open Banking initiative, exposing a range of APIs for account information and payment services. They need to implement a robust authorization system to protect these APIs.

Applying SABSA Layers:

1. **Contextual Layer:**
    - Business Driver: Comply with Open Banking regulations while protecting customer data
    - Stakeholders: Customers, Third-Party Providers (TPPs), Regulators
    - Risk: Unauthorized access to customer financial data
2. **Conceptual Layer:**
    - Security Principle: Zero Trust model for all API access
    - Conceptual Model: Attribute-Based Access Control (ABAC) for dynamic, context-aware decisions
3. **Logical Layer:**
    - Logical Components: Policy Administration Point (PAP), Policy Decision Point (PDP), Policy Enforcement Point (PEP)
    - Authorization Flow: API Gateway → Ping Authorize PDP → Resource Server
4. **Physical Layer:**
    - Technology Choice: Ping Authorize for centralized policy management and decision-making
    - Integration: API Gateway for initial request interception, Identity Provider for user authentication
5. **Component Layer:**
    - Ping Authorize Policy: Implement fine-grained policies based on user consent, TPP identity, and requested resource

- Attribute Sources: Customer consent database, TPP registry, transaction monitoring system

6. **Operational Layer:**
    - Monitoring: Real-time dashboard of authorization decisions and policy performance
    - Incident Response: Automated alerts for unusual access patterns or policy violations

SABSA Matrix Application:

Aspect	Application to SecureBank's API Authorization
Assets	Customer account data, payment initiation services
Motivation	Regulatory compliance, customer trust, secure innovation
Process	Account information retrieval, payment initiation, consent management
People	Customers, TPP developers, internal API team, compliance officers
Location	Various (TPP applications, customer devices, internal systems)
Time	24/7 availability with time-based access controls for certain operations

Implementation Highlights:

1. **Policy Design:**
    - Create a hierarchical policy structure in Ping Authorize, with global policies for common rules and specific policies for each API endpoint.
    - Implement dynamic consent checking, integrating with SecureBank's consent management system.
2. **Attribute Management:**
    - Configure Ping Authorize to retrieve TPP information from the Open Banking directory in real-time.
    - Implement a caching strategy for frequently used attributes to optimize performance.
3. **Integration:**
    - Integrate Ping Authorize with SecureBank's API gateway for seamless policy enforcement.
    - Set up secure attribute retrieval from various internal systems (customer database, transaction monitoring system, etc.).
4. **Monitoring and Audit

2.5 SABSA Matrix for API Authorization

The SABSA Matrix is a powerful tool that helps ensure all aspects of security architecture are addressed across all layers. Let's explore how it applies specifically to API authorization using Ping Authorize:

Layer	Assets	Motivation	Process	People	Location
Contextual	What API resources need protection?	Why do we need to secure our APIs?	What business processes involve API usage?	Who are the stakeholders in API security?	Where are our APIs accessed from?
Conceptual	How do we conceptualize API resources?	What are our API security goals?	What are the key processes in our authorization model?	What roles are involved in API authorization?	How do we conceptualiz API access locations?
Logical	How do we logically represent API resources?	How do we structure our security policies?	What are the logical steps in making an authorization decision?	How do we logically represent users and roles?	How do we logically define location in our policies?
Physical	How are API resources represented in Ping Authorize?	How are security policies implemented in Ping Authorize?	How does Ping Authorize process authorization requests?	How are user attributes stored and retrieved?	How does Ping Authorize determine the location of API requests?
Component	How are individual API endpoints configured?	How are specific policy rules defined?	How are policy decision processes configured?	How are user directories integrated?	How are geolocation services configured?
Operational	How are API resources monitored and managed?	How is policy effectiveness measured?	How are authorization processes monitored?	How are user access rights reviewed and updated?	How is location-based access monitored?

2.6 Applying SABSA to Ping Authorize Implementation

To effectively apply SABSA principles to a Ping Authorize implementation, consider the following guidelines:

2.6.1 Business Alignment

- Ensure that every aspect of the Ping Authorize implementation can be traced back to a specific business requirement or risk mitigation strategy.
- Use the SABSA matrix to map business drivers to specific Ping Authorize features and configurations.

2.6.2 Risk-Based Approach

- Conduct a thorough risk assessment of your API ecosystem.
- Use Ping Authorize's policy framework to implement controls that directly address identified risks.
- Prioritize the implementation of policies based on risk levels.

2.6.3 Layered Security

- Implement multiple layers of security using Ping Authorize in conjunction with other security controls (e.g., API gateways, threat protection systems).
- Use Ping Authorize's attribute-based access control to implement fine-grained authorization at different levels (API, resource, operation).

2.6.4 Measurable Outcomes

- Define clear, measurable objectives for your API authorization system.
- Use Ping Authorize's logging and reporting capabilities to gather data on policy effectiveness and performance.
- Regularly review metrics to ensure the authorization system is meeting its objectives.

2.6.5 Continuous Adaptation

- Leverage Ping Authorize's flexible policy framework to quickly adapt to changing business needs and emerging threats.
- Implement a regular review cycle for authorization policies, aligned with the SABSA lifecycle.

2.7 SABSA Lifecycle in API Authorization Context

The SABSA Lifecycle provides a continuous process for developing, implementing, and managing security architecture. Here's how it applies to API authorization:

2.7.1 Strategy and Planning

- Align API authorization strategy with overall business objectives.
- Identify key stakeholders and their requirements for API security.
- Conduct a risk assessment specific to API usage and data exposure.

2.7.2 Design

- Develop a conceptual model for API authorization using Ping Authorize.
- Create logical designs for policy structure and attribute usage.
- Design integration points with existing systems (e.g., identity providers, API gateways).

2.7.3 Implementation

- Deploy Ping Authorize in a phased approach, starting with non-critical APIs.
- Implement and test authorization policies.
- Integrate Ping Authorize with API gateways and other security controls.

2.7.4 Management and Measurement

- Establish monitoring and alerting for API authorization activities.
- Regularly review policy effectiveness and performance metrics.
- Conduct periodic audits of authorization policies and access patterns.

2.7.5 Continuous Improvement

- Gather feedback from API consumers and internal stakeholders.
- Stay updated on new features and capabilities of Ping Authorize.
- Regularly refine and optimize authorization policies based on operational data and emerging threats.

2.8 Challenges in Applying SABSA to API Authorization

While SABSA provides a comprehensive framework, there are challenges in applying it to API authorization:

1. **Complexity**: The comprehensive nature of SABSA can be overwhelming. Focus on the most relevant aspects for your API ecosystem.

2. **Rapidly Changing Environment**: APIs and their usage patterns can change quickly. Ensure your SABSA implementation allows for agility and rapid updates.
3. **Technical Depth**: SABSA requires a deep understanding of both security architecture and API technologies. Invest in training and potentially external expertise.
4. **Resource Intensity**: Fully implementing SABSA can be resource-intensive. Prioritize based on risk and business impact.
5. **Integration with Agile Methodologies**: Many API development teams use agile methodologies. Adapt SABSA processes to fit with agile workflows.

2.9 Best Practices for SABSA in API Authorization

1. **Start with Business Context**: Always begin with a clear understanding of business goals and risk appetite.
2. **Use Iterative Approach**: Apply SABSA in iterations, focusing on high-priority APIs first.
3. **Leverage Automation**: Use Ping Authorize's automation capabilities to implement and manage policies at scale.
4. **Educate Stakeholders**: Ensure all stakeholders understand the value and principles of the SABSA approach.
5. **Maintain Traceability**: Keep clear links between business requirements, risks, and implemented controls.
6. **Regular Reviews**: Conduct regular reviews of your SABSA implementation to ensure ongoing alignment with business needs.
7. **Integrate with DevSecOps**: Incorporate SABSA principles into your DevSecOps practices for API development and deployment.

2.10 Future Trends and SABSA

As API ecosystems evolve, consider how SABSA and Ping Authorize can adapt to future trends:

1. **Microservices and Serverless**: Adapt SABSA principles to highly distributed architectures.
2. **AI and Machine Learning**: Incorporate AI-driven decision making into your authorization policies.
3. **Zero Trust**: Align SABSA implementation with zero trust principles for API security.
4. **Blockchain and Decentralized Identity**: Consider how SABSA can be applied to decentralized systems and identities.
5. **Quantum Computing**: Prepare for the impact of quantum computing on cryptographic aspects of API security.

By thoroughly understanding and applying the SABSA framework to API authorization using Ping Authorize, organizations can create a robust, business-aligned security architecture that protects their critical API assets while enabling innovation and growth.