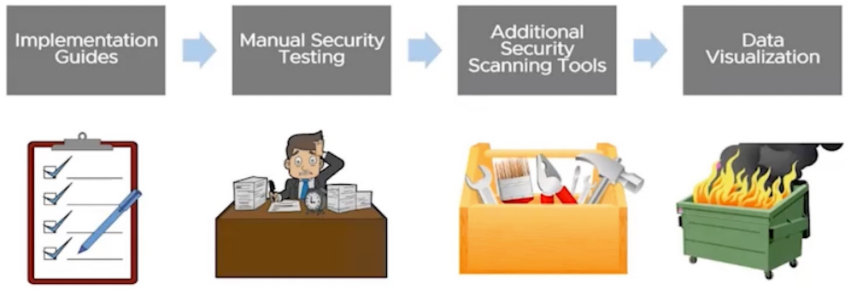
Software Architecture Assignment 1

Project Vulcan

Hemant Tiwari
2022MT93184
M. Tech in Software Engineering
BITS - WILP

Purpose of the system (Goal)

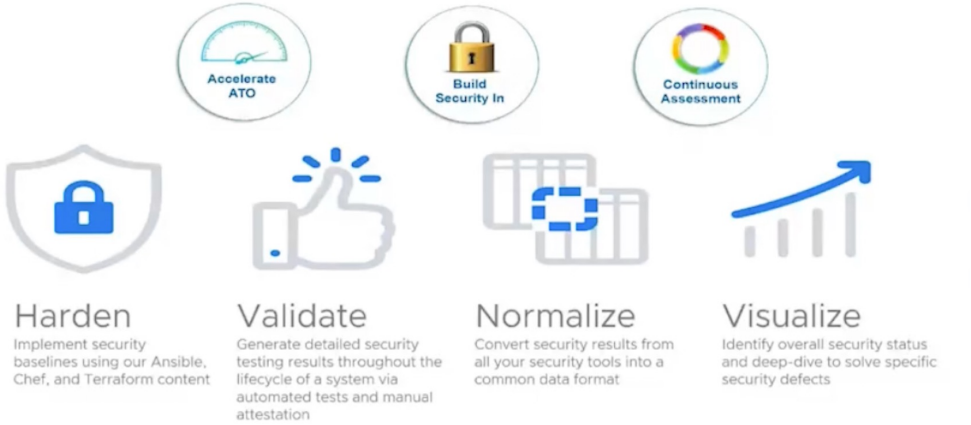
- A **Security Technical Implementation Guide (STIG)** is a configuration standard developed by **Department of Defense (DoD),** consisting of cybersecurity requirements for a specific product. The use of STIGs enables a methodology for securing protocols within networks, servers, computers, and logical designs to enhance overall security. These guides, when implemented, enhance security for software, hardware, physical and logical architectures to further reduce vulnerabilities.



Traditional Security validation Lifecycle

- Project Vulcan is a collaborative effort between ***VMware*** and ***MITRE*** to **modernize** the **STIG development process** by creating a tool that brings together the documentation side of assessing a product against DoD security requirements with automating this guidance as compliance as code.

- It helps in streamlining the process of creating STIGs and InSpec security compliance profiles. It models the STIG intent form and the process of aligning security controls from SRG items into actual STIG security controls. Vulcan also gives the option while aligning the security controls to insert inspec code and test across any type of system supported by InSpec.



Security Validation Lifecycle Using Vulcan

Purpose of the system (Goal)

COMPLIANCE AS CODE

- **Automation is tied to the source control.**
- **Content updates also update code.**
- **Changes automatically generated between releases as a detailed diff view.**

GOVERNANCE

- **Scale content generation to stakeholders.**
- **Approval process**
- **Track changes and revert**
- **Release process**
- **Permissions model to support multiple projects and roles**

EFFICIENCY

- **Artifact generation automated (XCCDF, Inspec XLS, Revision History)**
- **Content reuse of common components**
- **STIG ID generation**
- **Import existing content in spreadsheets**
- **Handle adding controls as needed.**
- **Associate requirements met by other controls**
- **SRG revision updates**

USER EXPERIENCE

- **Functional replacement for spreadsheets**
- **Sort and Filter controls by various fields Searchable**
- **Embedded guidance**
- **Spell check**
- **Comment History**

STRATEGIC PRIORITIES

- Support STIG project engagement with DISA
- Enable other compliance needs (FedRAMP, ILA/5/6)

Key requirements of the System – Functional & Non-Functional

- Project Vulcan can be divided into three major views.
    - **Projects** - Are the high-level construct to initiate a STIG development effort for a product. Projects have Members added to them with various roles which apply to all their components.
    - **Components** - Components have a 1:1 relationship with a specific component of a product and it's applicable SRG. When a component is completed, it is "released" in Vulcan and no longer editable and enables other workflows such as artifact generation and publication.
    - **Controls** - Controls are the lowest level construct and for a component there is one control for every requirement in an SRG although some requirements can be split into multiple controls if necessary to break out different configurable items.
- There are various roles a user can hold in Vulcan depending on the type of work they need to do inside a project.
    - **Admin -** An admin for a project can create/delete/release components, add members and update roles, and edit and approve controls within components.
    - **Reviewer -** A reviewer can edit and approve controls that are marked for review.
    - **Author -** An author can update status, edit fields, write check and fix text, and submit controls for review.
    - **Viewer -** A viewer is essentially a read only role for just viewing components and controls.

Key requirements of the System – Functional & Non-Functional

From the **Project view** following are the requirements,

- Add members to a project
- Create/delete/export/release/copy components
- Import existing components from a spreadsheet
- Download XLSX for released components
- Add project info as metadata
- Open components for viewing or editing

From the **Component view** following are the requirements,

- View controls and their status
- Enter edit mode
- Release a component
- Add/update component metadata
- Add/update custom fields
- Add members to a component
- Filter controls by control status or review status
- Search controls

From the **Controls view** following are the requirements,

- Edit Control Status.
- Submit the control for review.
- Add comments on the controls.
- Mark a specific control duplicate.
- Reviewers can then approve a control
- Reviewer post review can lock a control from further editing
- Reviewer post review can request changes and send it back for further editing.
- Revert changes to a control under the revision history section.

Utility tree of Architecturally Significant Requirements (ASR)

Quality Attribute	Attribute Refinement	Scenario	Business	Architecture Impact
Security	Accessibility	As Data/Sheets contains sensitive information regarding the application being hardened. Therefore, it should only be accessible by the people who have been granted access to the sheets.	High	High
Availability	Service Availability	System should support 99.999% Availability	High	High
Performance	Scalability	System should support big (larger size) data ingestion and should be scalable further.	High	High
Usability	User Experience	The UI should be user friendly.	High	Low
Modifiability	Criteria specification	User should be able to modify use cases.	Low	Medium
Interoperability	Notification	The system should send real time notification to the admins if any changes occurs.	High	Medium
Maintainability	Easy Operation & Maintenance	The system should be easy to operate and maintain in production.	Medium	High
Maintainability	Data backup	System should keep back up of complete system data including configuration and logs.	High	High

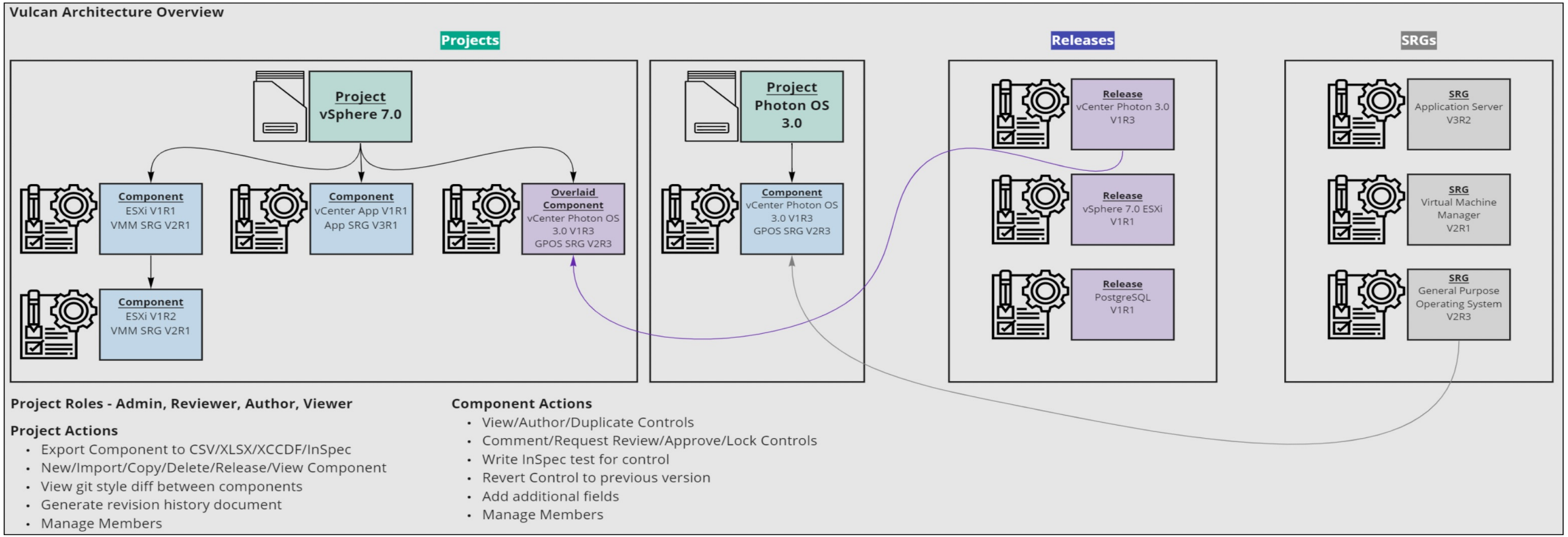
Tactics used to achieve the top 5 ASRs

Quality Attribute	Scenarios	Tactics
Security	As Data/Sheets contains sensitive information regarding the application being hardened. Therefore, it should only be accessible by the people who have been granted access to the sheets.	The Vulcan is installed in a completely isolated private cloud.
		The URL is accessible only via Organization VPN.
		No link is made between the Origination Level LDAP Servers.
		Only Admins can provide access to the Users.
		No users will have access to more than two project at the same time.
		There can be no more than two admins.
Maintainability	System should keep back up of complete system data including configuration and logs.	Stimulated Disaster and Recovery (SDR) is performed to check the vulnerability of each components.
		Based on the SDR backup of Logs, Database, Namespaces, etc. is taken every 30 minutes and is stored in a different lab.

Tactics used to achieve the top 5 ASRs

Quality Attribute	Scenarios	Tactics
Interoperability	The system should send real time notification to the admins if any changes occurs.	Develop use cases to trigger email and SMS alerts to admins for critical events.
Performance	System should support big (larger size) data ingestion and should be scalable further.	Using of Could Storage
Availability	System should support 99.999% Availability	Using Tanzu and Kubernetes for Deployment.
		Deploy server in HA mode
		99.999% Availability based on Distributed Architecture and Active-Active GDR

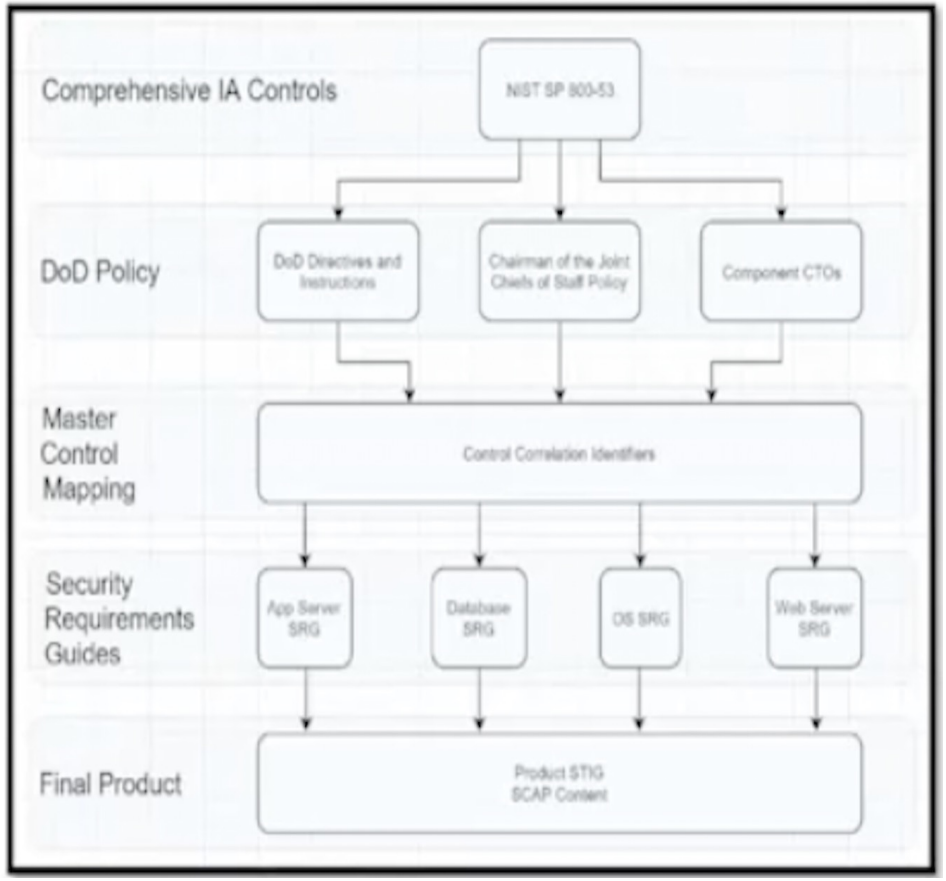
Software Architecture diagram

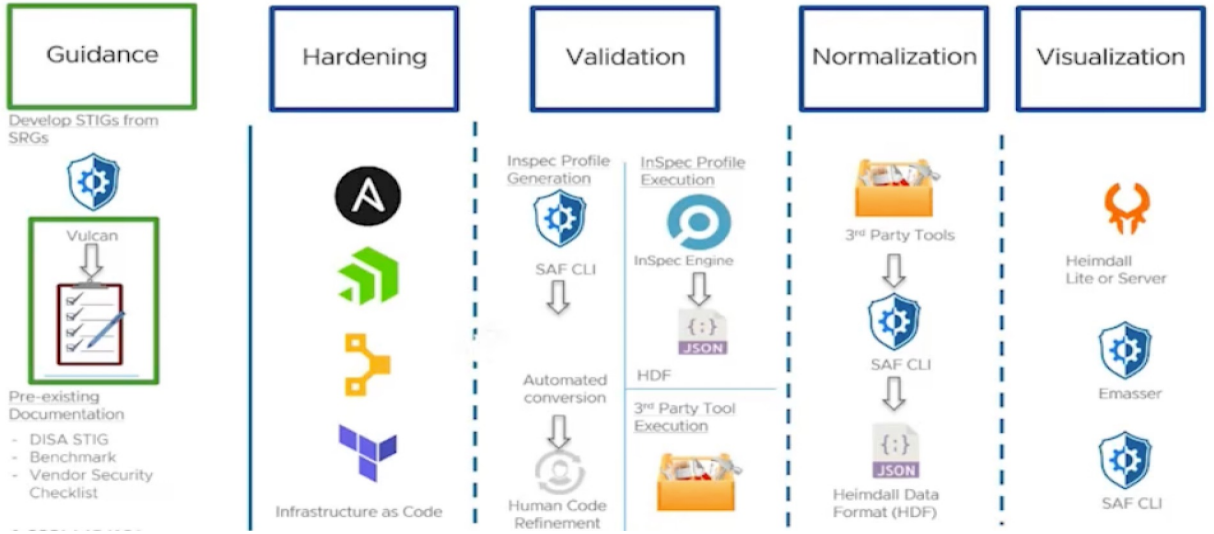


Note – Due to Security and Confidentiality can not provide any other architectural diagrams.

Description of How the System Works

- Model the STIG creation process between the creator(vendor) and the approver(sponsor).
- Write and test InSpec code on a local system, or across SSH, AWS, and Docker.
- Easily view the progress on what the status is of each control.
- Communicate through the application to make the best decisions on controls.
- Confidential data in the database is encrypted using symmetric encryption.
- Authenticate via the local server.





Key Learnings

- Importance of Architectural Diagrams, ASRs and Utility tree, and Functional and Non-Functional Requirements in Software Architecture.
- Learnt about key requirements and how they are categorized into functional and non-functional requirements.
- Understood the different types of diagrams and how they play a role in software architecture.
- Architecturally Significant Requirements documentation.

Thank You