Implementation of DevSecOps for D2D

Disclaimer: This presentation is about where we’re going, not where we are. Some pieces still need to be built out

Agenda

- How to get started (Prerequisites)?
- Who does what (teams/roles)?
- Baseline Tools
- Development Methodology
- Security Assessment and Authorization Process
- Operations
- What is DevSecOps?
- Infrastructure as Code (IaC) build process
- Continuous Integration and Continuous Deployment (CI/CD)
- DevSecOps - Proposed Roles and Responsibilities
- Questions?

How to get started (Prerequisites)?

- **Work with BSP/IDI to request new AWS accounts (Dev/Test/Prod)**
- **Work with BSP/IDI to create one VPC for each account and configure VGW to ensure communication between the new AWS accounts and link it to master account. Additional VGW configuration would be required to connect to NetOps VPC, Mgmt VPC, and SecOps VPC**
- **Open ServiceNow tickets to create the below domain accounts:**
  - **Short Name Account (SNA)** for ENT Domain (one ticket per person via catalog)
  - **BSP Short Name Account for Jump Server Access**. (one generic ticket list all users, this will be assigned to BSP Support Team). This provides access to the BSP environment for any Mode
  - **Create Domain Group** for your team (one ticket via general ticket, then list all users, this should be assigned to Directory Services). This identifies your group in the Domain
  - Create custom IAM policy and include all IAM access/privileges that may be required to provision AWS resources

Who Does What?

- **BSP Team** - BSP team can provide the initial guidance and support to establish AWS accounts and access to BSP cloud environment
- **IDI** - IDI team can create the IAM policy accounts. They act as integrators with other GSA support teams to facilitate the cloud environment for the migration team (However, IDI's role could change soon)
- **SecOps** - Security Operations (SecOps) team provides the policy guidance needed to successfully complete the security authorization process (e.g., LATO, ATO) and will coordinate with the SecEng team to approve the documentation provided by the application team through control assessments
- **NetOps** - Network Operations (NetOps) provides the appropriate network support requirements to migrate applications to the cloud. This includes opening of ports, whitelisting IP's for access to internet and access to appropriate repos for application updates/patches, access to other servers in the GSA network, establishing subnets (public and private) within the VPC CIDR, DNS, and configure inter VPC connections and peering (if required).
- **SecEng** - Security Engineering (SecEng) works in tandem with SecOps and will provide appropriate guidance and support to review and approve the architecture diagrams for the applications being migrated to cloud. This includes Security Groups (SG's), Network Access Control List (NACL), SSL ports (for secure transfer of data), encryption for "data at rest" and "data in transit", and ensure all security controls are incorporated in the final application architecture. This team will coordinate with the SecOps team to approve the documentation provided by the application through control assessments

Baseline Tools

- JIRA (Agile sprint planning)
- CloudFormation Templates and/or Terraform
- Ansible
- GitHub
- Slack
- Jenkins
- Docker
- Google Drive
- ServiceNow

Development Methodology

- The team starts off with sprint planning meeting, where all stakeholders (including scrum master, development team, project sponsor, security, and other project stakeholders meet to discuss the various tasks at hand and finalize the stories/tasks and then are prioritized for deliverables. These deliverables are time boxed for the two-week sprint and remaining tasks are placed in backlog
- **GitHub -** Git is a distributed version control system
  - Git repository is the source of truth in terms of code and the state of the system
  - The \`master\` branch is in a working state at all times (Feature branches can always be created off of \`master\`, meaning team members aren't blocked waiting for \`master\` to be fixed)
  - All required/desired technical changes exist as issues in GitHub repo
  - Commit messages explain the change(s) in a meaningful way
  - The repository contains up-to-date instructions on how to deploy the system, comprehensible by someone who isn't familiar with it
  - **Pull Requests**
    - All changes to the system are done on "feature branches", and submitted through pull requests
    - Pull requests are reviewed and merged by someone with relevant experience who isn't the author
    - Team members are encouraged to submit pull requests early

Development Methodology...contd.

- **Open Source Guidelines**
  - There is no (unencrypted) sensitive information in the repository
  - What this means is, if GitHub getting hacked, this wouldn't mean our system could be compromised
  - All code developed by/for the D2D team is open source
  - What this means, other teams at GSA and beyond can benefit by studying/referencing/reusing/contributing to our documentation and code
- **Portability**
  - Given sufficient permissions (appropriate IAM policies), the system/code can be deployed in any AWS account
  - The system will work with either the custom AMI's developed by the IDI team, or the equivalent (off-the-shelf) AMI from the AWS Marketplace
- **AMI's**
  - The development team can get a marketplace AMI and develop a custom AMI by deploying the security hardened scripts (currently in shell scripts format, but soon to be available in Ansible playbooks format)
  - IDI will create a centralized Git to store and maintain baseline security hardened AMI's. These AMI's will be made available for development teams to use in their individual AWS accounts. The development teams can take these baseline AMI's and customize them (adding application specific configuration items) to create their own custom AMI's

Development Methodology...contd.

- **AWS Cloud Formation Templates (IaC)**
  - CloudFormation is AWS-specific and can be used to provision just about any type of AWS service
  - AWS CloudFormation gives developers and DevOps team an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion
  - Build and use AWS CloudFormation templates (JSON scripts) to build Infrastructure-as-a-Service (IaaS) code and deploy the code to build AWS services in the environment
- **Terraform (IaC)**
  - Terraform uses HCL (HashiCorp Configuration Language) as its template language. HCL is compatible with JSON and, if desired, we can write Terraform templates in JSON
  - Is a cloud-agnostic tool which enables the provisioning of hybrid-cloud infrastructures with a single tool
  - Use Case: We can use **CloudFlare DNS** along with an AWS infrastructure. Terraform will allow us to provision both in the same template with the same tool
  - Another advantage of Terraform is its separate planning step. Running terraform plan generates an execution plan that will show exactly what Terraform will do when you apply the template to your infrastructure and in what order
- **Ansible (CM)**
  - Ansible is a configuration and orchestration management tool. It supports agentless architecture, and has ability to interact easily with the ever-changing, scaling, and dynamic AWS architecture

Development Methodology...contd.

- **Ansible**
    - We used Ansible to integrate our AWS CloudFormation templates (IaC) and the application code to deploy our infrastructure and applications with one or a few commands in AWS cloud environment
    - Once AWS-based application environments are described with Ansible, you can deploy them again and again, easily scaling out to 100s or 1000s of instances across multiple regions, with the same results each and every time

- **Underlying Principles for Development Methodology**
    - Programmable and repeatable infrastructure for various projects across GSA of all sizes
    - A template based-tool set used to declaratively define and build infrastructure resources
    - Enable project teams to use standard code for self provisioning, and allow application developers to apply software development tools such as version control systems (VCS), automated testing libraries, and deployment orchestration to manage infrastructure
    - Application owners are able to define, provision, and manage the infrastructure resources they need, without needing operations staff to do it for them
    - Improvements are made and deployed continuously (CI/CD), rather than done through expensive and risky “big bang” projects
    - Testing, provisioning, and management of infrastructure are done as part of automated pipeline
    - Infrastructure is deployable as individual components

Security Assessment and Authorization Process

- The LATO process requires the documentation of a System Security Plan (SSP), review and approval of the system security architecture, and assessment of the security controls
- The premise of the approach is that the Security Operations (SecOps) and Security Engineering (SecEng) teams would be engaged to perform incremental reviews and approval of documentation and controls, as well as incremental scans (e.g. scans of AMIs and mitigation of high/critical findings would be performed first along with individual application scans as they are deployed to the environment)
- This is made possible with an established dedicated cross-functional teaming structure established along with the agile scrum development methodology. The security team is actively involved in the sprint planning sessions and will have stories built in each sprint to perform incremental review/approval of security controls for the automation code developed in a particular sprint
- The development team and security team will need to closely collaborate to map technical solutions to security controls and align deliverables
- The development team and security team will manage the control assessment and document review schedule
- The security team will have a task to review and approve a “pull request” before it’s merged. This will ensure security buy-in to the code being deployed (This process could change soon when SCS and DCS is made available)

Security Assessment and Authorization Process

- **Security Templates**
  - **Application Authorization Traceability Matrix:** Used to track incremental updates to the System Security Plan and associated security documentation
  - **Security Controls Mapping to Migration Component:** Completed per sprint at sprint planning sessions. This document shows the mapping of components to impacted controls that are planned to be deployed during a given sprint
  - **Control Assessment Schedule:** High level schedule showing the anticipated security control assessment schedule by component instead of traditionally by control. This schedule is based upon the project schedule and when components are planned to be deployed.
  - **Consolidated Security Findings Tracking:** Consolidated tracking sheet that shows the security scanning results as well as the adjudication or remediation for each finding

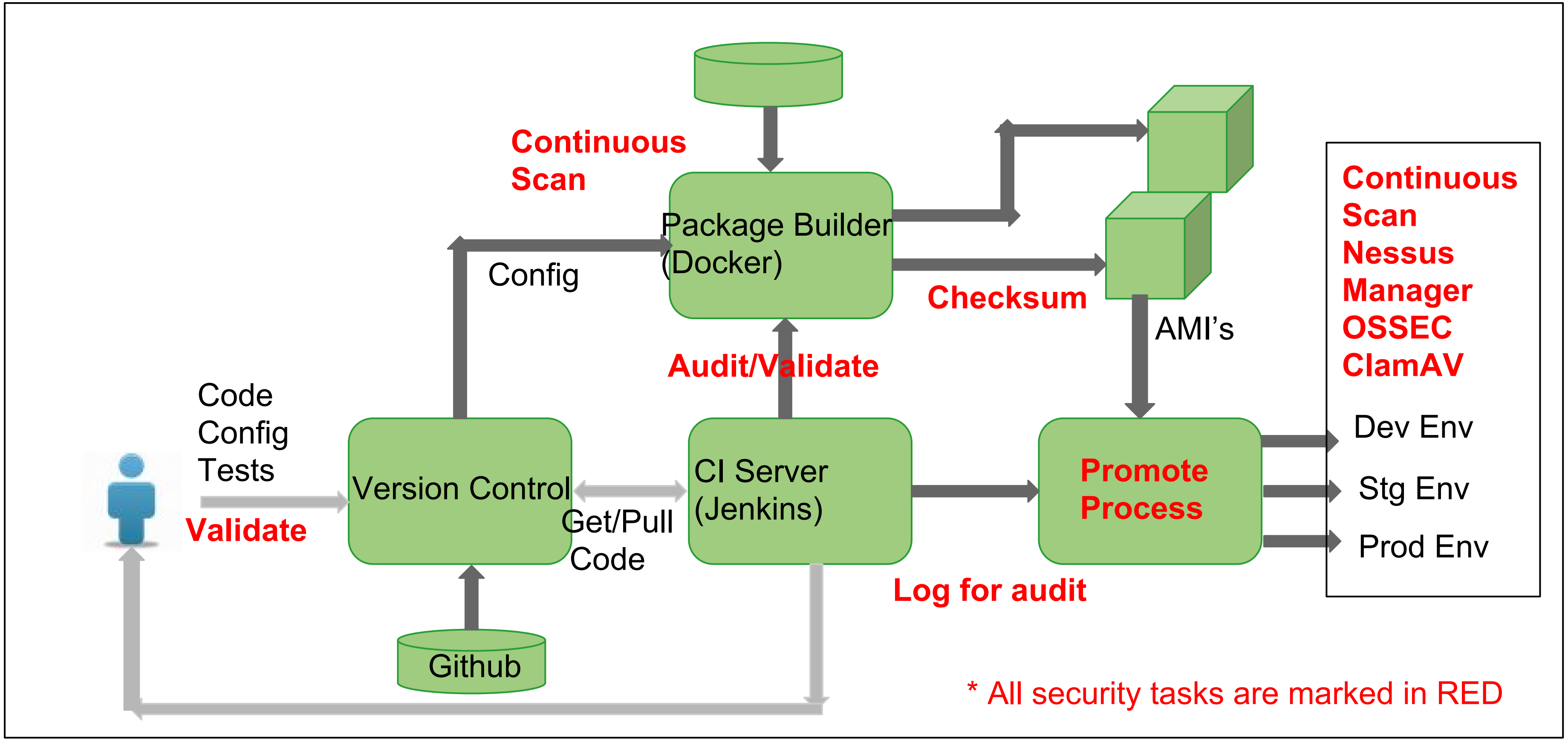
Operations

- **Monitoring and Orchestration layer** has a goal to provide operational, and security alerts, analysis and growth sustainment, while providing continuous improvements. We currently have CloudWatch and SNS configured for monitoring and alerting
- **Content layer** represent analytical models, Tableau workbooks, MSTR projects, Alfresco Document types, etc. that are artifacts developed by Analyst and Content Managers and deployed in particular tools via defined workflows. Interface with management tier to enable easier deployment and management of content
- **Management layer** has the goal of abstracting away both the tooling (components) and infrastructure layers into one common workflow for turning application code into a running application. An example end goal would be to just package the application — potentially as a Docker container, jar or binary — and have it be able to run on any cloud infrastructure. Ideally, no knowledge of the infrastructure itself is necessary and all the logic is held at the application level
- **Components layer** is comprised of technologies that make access to and configuration of the infrastructure layer an easier process and facilitate automated deployments of D2D components and data. Layer’s enablement tools still require functional knowledge of the underlying infrastructure. Examples of these tools include open source tools by Puppet, Chef, Jenkins, etc.
- **Infrastructure layer** is comprised of technology that provides virtualization functionality to physical hardware to make compute, network and storage easily accessible. Example technologies are Amazon’s EC2, RDS, S3, etc.
- **Patch Management** is done through Ansible playbooks. The new patches are called (YUM update for Linux and Windows update for Windows) from Ansible code and deployed through “hosts.yml” file (which has a list of hosts in the environment)

What is DevSecOps?

Attribute	Key Elements
High trust, High performance culture	Unified mission, aligned incentives across multiple functional teams, high quality of work life
Highly automated processes, mature deployment pipelines	Technical phases of projects supported by common tools and automation processes, collaboration replaces handoffs, codebase/IT infrastructure is agile and functional by default. Security team is involved in the automation process and helps map security controls to build automation capabilities
Continuous delivery of software and IT value	Features, sprints (small projects), and delivery follows a regular, iterative flow. Cycle time is short (2 weeks), workflow favors small frequent changes inclusive of security team’s tasks
Commitment to continuous learning and improvement	Disciplined feedback loops (retro sessions) quickly travel back upstream for inclusion. Shared knowledge repositories

Continuous Integration/Continuous Deployment



* All security tasks are marked in RED

CI/CD & DevSecOps - Run book (Automation Work Flow)

Build

- **JIRA (Agile)** – Scrum/Kanban boards for sprint planning (stories/tasks), tasks priorities, track/report progress
- **Cloud Formation and/or Terraform** – JSON scripting for IaaS components
- **Ansible** – Configuration Management for cloud provisioning
- **Github** – Centralized code repository (version control, collaborative code review (branching, merge, & commits), Pull/Push requests, integrated issue tracking
- **Slack** – Team communication/collaboration

Test

- **Jenkins – Continuous Integration (CI) for Application Development**

Work Flow: Developers commit changes to Git repository → Github triggers Jenkins job (based on cronjob settings) and checkout the code → Jenkins runs the build and creates packages → Jenkins runs unit and integration tests and test results are stored → Jenkins sends build notification (with version control)to the development and deployment teams

Deploy

- **Jenkins – Continuous Deployment (CD) for application deployment**

Work Flow: Using the Jenkins deployment dashboard, select the latest version of the application software and select the environment (DEV/TEST/PROD) to deploy the application → The Jenkins job will use the parameters to download the correct artifact from the repository → The Jenkins job will download the deployment scripts from the repository to provision the EC2 hosts with the application install

Operations

AMI - Automation to build standard security compliant OS images, version control for security updates/patches

Security Tools – Automation to deploy security tools on EC2 hosts. Tools like Nessus (vulnerability checks), OSSEC (HIDS), and ClamAV (Anti virus)

Monitoring Tools – SolarWinds (Server/Application monitoring, Event management), CloudWatch

Logging - CloudWatch Logs

DevSecOps - Proposed Roles & Responsibilities

	Process(s)	Technology
Dev Team	Application Development Sprint management	Application code Pipeline (**Application coding language and Jenkins**) **Docker container** builds for application deployments **GitHub** (app. code/version control)
Security Team	Log capture/monitoring Incremental LATO/ATO process Static Code Analysis	**Nessus Manager** **OSSEC Manager** **ClamAV** **Fortify** **Docker UCP**
Operation Team	Managing AMI lifecycle Patch Management Managing AWS IAM Accounts Managing Network Definitions Financial (chargeback) management	**CF templates and/or Terraform, Ansible** **Jenkins** (for IaaC pipeline) **Docker container** (for infrastructure and other AWS services deployments) **Github** (IaaC code/version control)

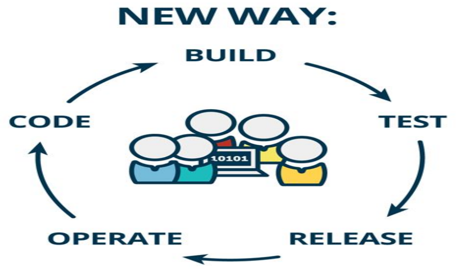
Questions?

BACKUP SLIDES

Why DevSecOps?

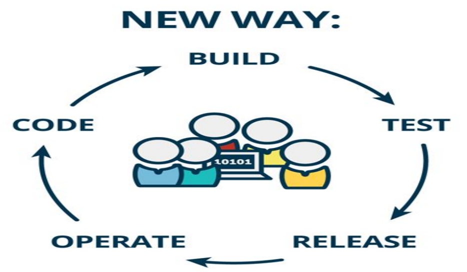
- Automation - deployments are built through code
- Agility - treat infrastructure as code, meaning changes can be rapid and incremental
- Collaboration - Continual collaboration between information security, application development, and IT operations teams. Having all three teams immersed in all development and deployment activities makes it easier for information security team to integrate controls into the deployment pipeline without causing delays or creating issues by implementing security controls after systems are already running (helps facilitate incremental LATO approval process)

GOAL: Collaborative, automated, 14-day delivery Processes (smaller deliveries with full ATO)



Why IaaC?

- Automation - Infrastructure deployments are built through code
- Agility - treat infrastructure as code, meaning changes can be rapid and incremental
- Self Provisioning - Enable infrastructure self provisioning for application teams
- IaaC as a complement to DevOps - IaaC delivers the infrastructure environments with the rapidity, flexibility, fidelity and inter-team portability needed to make DevOps work
- Simplicity - removing the complexity of human intervention and replace with repeatable, standardized pipelines for infrastructure deployments



How CI/CD & DevSecOps Helps us?

- Automated and autonomous security
- Implement the control segments to validate and audit code and artifacts as part of CI/CD process
- Assurance that our architecture and deployment process is validated against GSA wide security policies
- Avoid infrastructure/application failure at the deployment stage due to non-compliance of security configuration
- Match DevOps “pace of innovation” and shorten the LATO approval process (facilitate incremental LATO approvals)
- Continuous auditing & alerting
- Security at scale