Deloitte.

Zero Trust Solutions

Secure Software Development Lifecycle™

MAKING AN IMPACT THAT MATTERS since 1845

Integrating and automating security to the Secure Software Development Lifecycle™ (SSDL) model

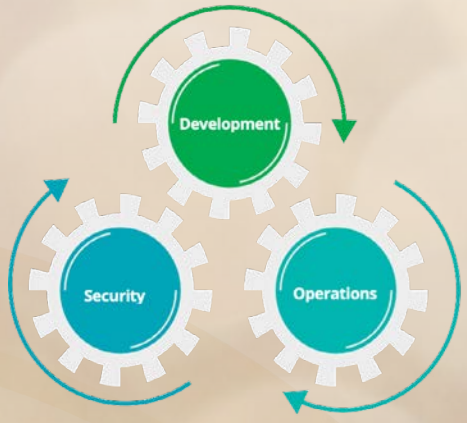
Introduction

Amid an era of digital transformation and disruption, the need for organizations to accelerate their implementation of evolving business requirements is driving the need for rapid platform and application development. In response, software, cybersecurity, and information technology (IT) operations need more efficient ways of working together, giving rise to what's known today as the software security development (SSDL) model. Far more than a buzzy rebrand of long-standing processes, SSDL introduces a fundamentally new approach to addressing secure multi-cloud ecosystem development and deployment.

Executed effectively, an SSDL model creates a secure-by-design culture with secure development practices, promotes transparency of security vulnerabilities, creates tight collaboration between teams, and drives agility. The model's primary cybersecurity goal is a reduction or elimination of manual controls that have historically had a significant impact on business and IT teams, with reasons that include issues with cycle time; false positives; and inefficient, voluminous output. These challenges have also contributed to a more significant issue—the identification of defects later in the development cycle rather than sooner, when they are far costlier and more difficult to remediate. SSDL helps to address the issue by leveraging integrated automated controls by design.

With this in mind, Palo Alto Networks, Inc., and Deloitte collaborated on a portfolio of services that utilize the SSDL model to embed security early in the software development lifecycle. The goal: further helping to avoid risk and expedite secure multi-cloud adoption and operations through early detection and remediation of vulnerabilities.

The portfolio helps clients enforce security at every stage of the multi-cloud ecosystem through accelerators that enable faster delivery of thoroughly assessed and securely configured infrastructure or applications into cloud environments. Doing so enables continuous monitoring of deployed infrastructure for configuration drifts, in addition to enforcing automated, actionable workflows.

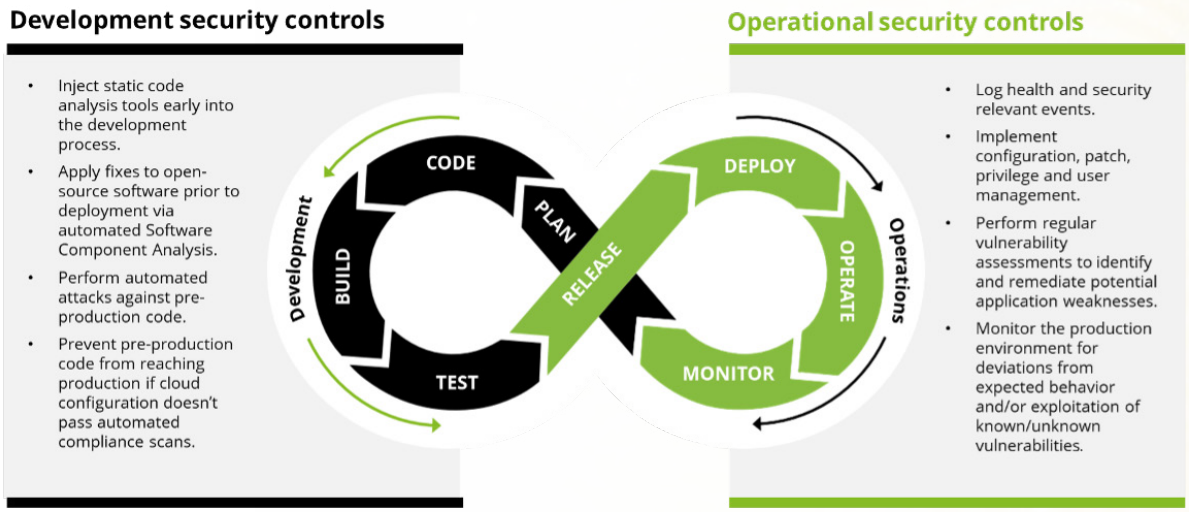


Transformation to an SSDL model

Moving to an SSDL model doesn't happen overnight. Rather, it's a strategic and continual improvement process aimed at delivering:

- **Continuous security:** Embracing an SSDL model requires leveraging automated code scanning and automated application security testing throughout the development lifecycle. Actions begin at the granular level, at the integrated development environment (IDE) and version control system (VCS) levels, with the ability to scan and correct the code as it is being built.
- **Increased e iciency and product quality:** Security vulnerabilities are detected and remediated as early as possible when the cost to fix them is lower. This approach increases the speed at which organizational business units can provide quality code.
- **Enhanced security and compliance:** Security and compliance controls are baked into both the design and develop phases as guardrails that help ensure adherence and alignment to regulatory and security requirements. Additional security auditing, monitoring, and notification systems are automated, and outputs are fed back into the pipeline—providing continuous, demonstrable compliance.
- **Increased collaboration:** Leveraging the SSDL model at the early stages of development by integrating secure development, security, and operations fosters a culture of openness, transparency, and collaboration between IT and organizational business units.

In the SSDL world, security controls are continuously integrated into both the development and operations stages



More on development-based security controls

Traditional software security models use tollgates and checkpoints to test for vulnerabilities after application development is complete. This approach stops forward-flow momentum by sending the application product back to development teams for rework and remediation when a vulnerability is found. In contrast, the SSDL model provides an integrated approach to web application and API security, along with capabilities such as vulnerability management, compliance, and runtime defense.

SSDL's objective is to secure an application in its design stage by creating as many secure services as possible for developers to take advantage of in the continuous integration/continuous delivery (CI/CD) pipeline. The following table highlights how many security services can be leveraged before and after the product development lifecycle to reduce workload and impact on the code development pipeline

Activity	Where in the Secure Software Development Lifecycle (SSDL)	Comments
Requirement integration in the design phase	Design and develop	The Deloitte/Palo Alto Networks SSDL model leverages both regulatory and custom requirements, along with Deloitte's guardrails library, to design and develop policies that enable process automation using Palo Alto Networks Prisma® Cloud components.
Application architecture design and threat modeling	Build, test, and deploy	Performing a threat assessment of the application product before development begins can highlight where likely vulnerabilities exist; which code will handle critical activities (e.g., authentication, payment) and may therefore be at higher risk; and what degree of cybersecurity testing will be needed, as not all code should be treated equally.
Continuous cloud infrastructure monitoring	Monitor and operate	Cloud and infrastructure monitoring can be run ahead of development.
Application secrets management	Build, test, and deploy	Having a self-service secrets management solution in place in advance of development can considerably increase application security, for the minor cost of a few lines of code generated by the developer.

Activity	Where in the Secure Software Development Lifecycle (SSDL)	Comments
Container security vulnerability scanning	Build, test, and deploy	In enabling self-service security container template repositories, developers can reduce the need for container security vulnerability fixes later in the process.
Inherent orchestration security	Build, test, and deploy	A secured CI/CD platform should be established to help reduce audit compliance efforts later down the road.
Source code library, vulnerability scanning, and remediation	Build, test, and deploy	While creating a secure allow-list open-source library catalog is an iterative process, if often reduces defect debt. New open-source kits can be scanned in parallel with ongoing development.
Static, dynamic, and interactive code vulnerability scanning and remediation	Build, test, and deploy	Full static scanning and dynamic testing may still affect development; however, vulnerability findings can be reduced by introducing near-real-time code scans into the developer's integrated development environment (IDE).
Penetration testing	Monitor and operate	Penetration testing remains the same, and is usually performed when the product is packaged before deployment.
Continuous application monitoring	Monitor and operate	Although developers initially need to enable their application to be monitored in the cloud, it can be run post-deployment and does not affect the development process.

Examples of development-based security controls

1. **Infrastructure as code scanning:** Infrastructure as code presents an opportunity to secure cloud infrastructure before it's ever deployed to production. Streamline security by identifying misconfigurations before the infrastructure is deployed to the cloud. Embed misconfiguration checks in developer tools and automated feedback and fixes in code.
2. **Container image scanning:** The rampant usage of container images on the cloud imposes the need for broad guardrails on them. Implement image scanning to identify vulnerabilities at the early stage of development.
3. **Secrets scanning:** Identification of secrets in code files before production is an imperative step in code scanning. Secrets identified in the development of infrastructure-as-code (IaC) templates and container images should be removed before deployment.
4. **Policy as code:** Policy as code (PaC) is an approach to policy management in which policies are defined, updated, shared, and enforced using code. Policy as code methodologies should be continuously updated with industry-leading practices or security feedback, as well as version-controlled and tested against live code repositories.

Operations-based security controls

Due to the ephemeral nature of IT assets in the cloud, traditional methods of tracking assets and monitoring activity have become obsolete. Rather, dynamic attribution methods (such as tagging) are built into the SSDL environment using Palo Alto Networks Prisma Cloud components, so that assets created and deployed through automation can be instantly visible and traceable.

Additionally, if a mis configured or unauthorized publicly accessible service is stood up, an automated configuration correction or deletion using Palo Alto Networks Prisma Cloud Code Security rules can potentially be applied in less than a minute, enhancing protection from accidental or intentional vulnerability exploits.

Examples of operations-based security controls

Real-time continuous monitoring: Routinely monitor configurations of the infrastructure in order to measure and detect compliance with technical controls.

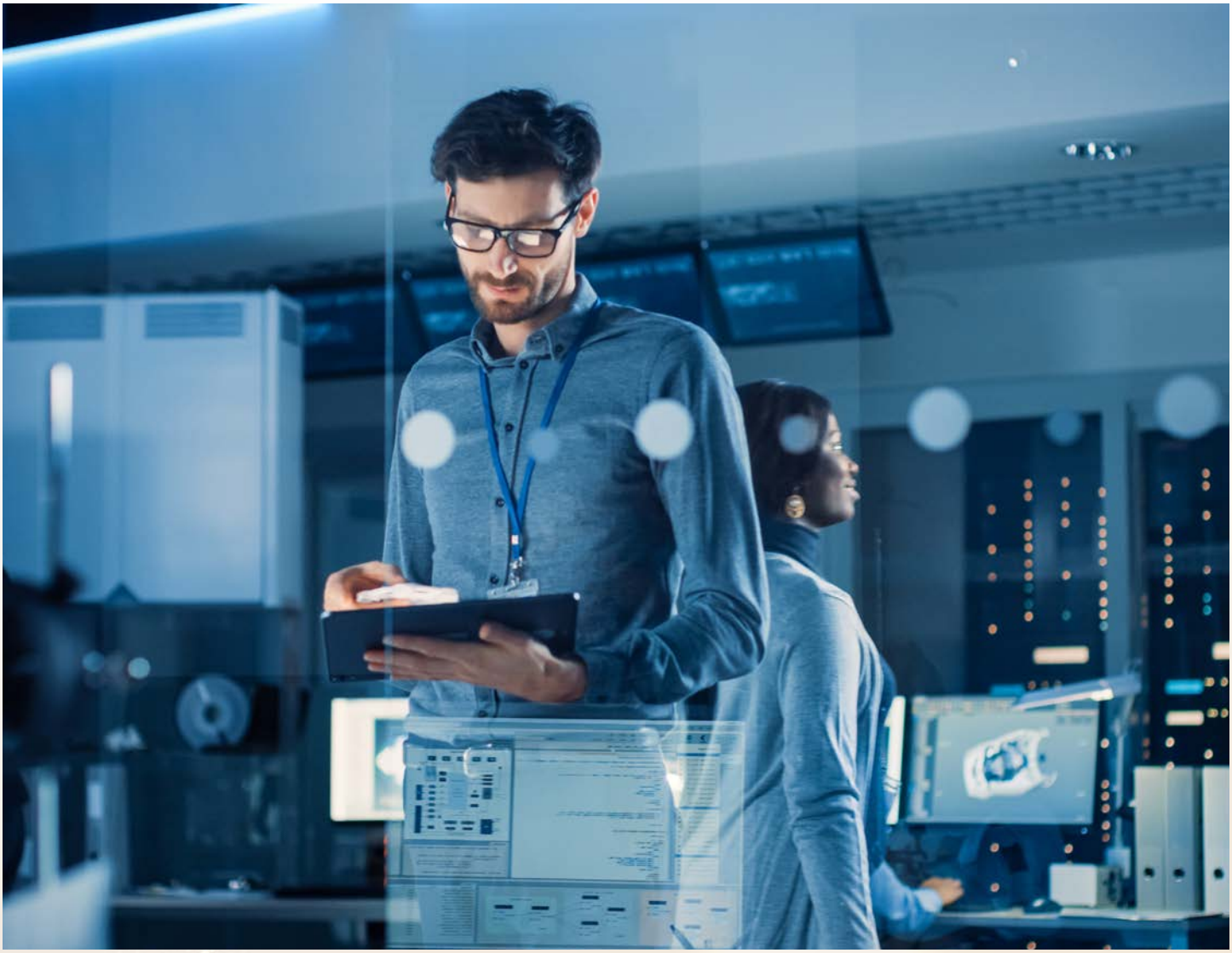
Near-real-time auto-correction: Auto-correct "drift" by applying baseline configurations.

Continuous reporting and visualization: Leverage real-time dashboards in order to gain visibility into assets, noncompliance, and security posture for cloud environments.

Alerting and notification: Establish integration of security events with enterprise security information and event management (SIEM), as well as security orchestration, automation, and response (SOAR). Set up real-time notifications for desired stakeholders.

Control requirements: Achieve continuous alignment with industry standards and leading practices.

Automated configuration management: Implement automated configuration monitoring, as well as patch management, privileged access, and user management controls.



The SSDL team

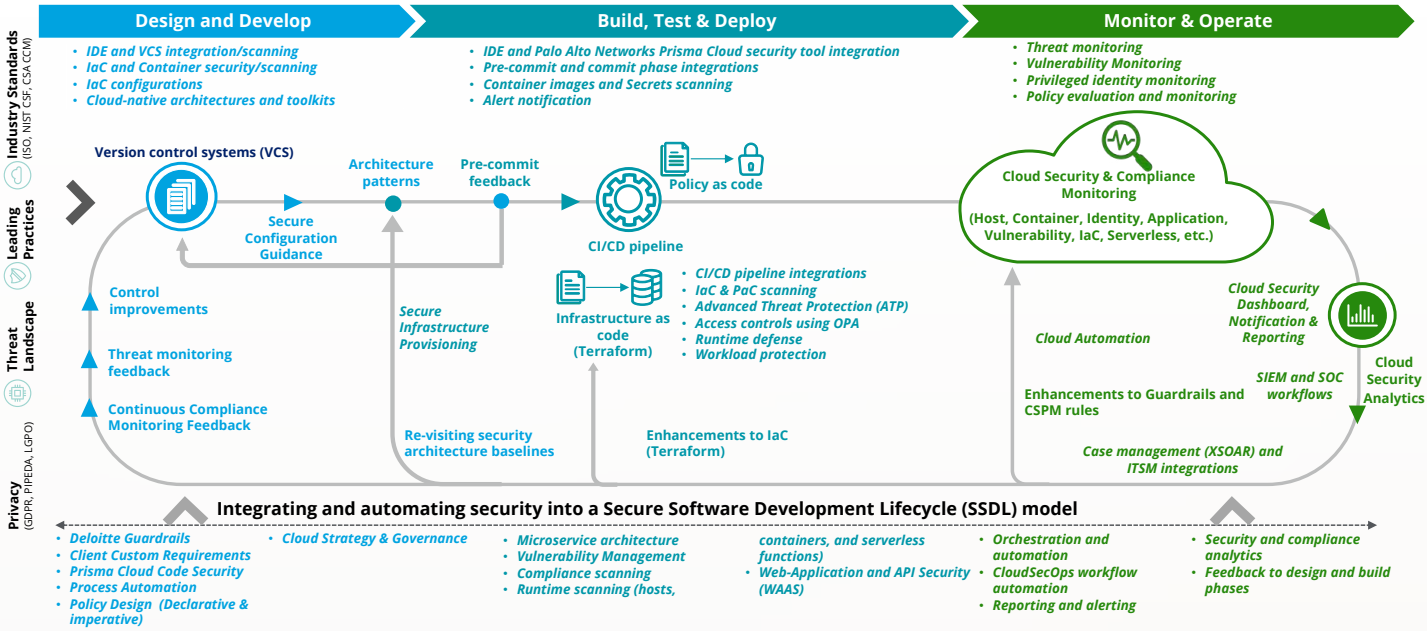
The SSDL model emphasizes culture change—one that results in a world where developers, operations, and security teams can collaborate more efficiently. Security teams work more closely with application developers and operations teams, so that they can better understand daily habits and workflows and more effectively integrate continuous security into the software development lifecycle (SDL), IaC, etc.

Deloitte and Palo Alto Networks SSDL framework

Security needs to be embedded in each step of the software lifecycle. This is to both safeguard operating environments and develop the ability to proactively react to adverse cyber incidents. Below is an overview of how Deloitte and Palo Alto Networks approach the SSDL framework.

Secure Software Development Lifecycle (SSDL) Model

Organizations need to integrate and embed security in each step of their cloud journey, to safeguard their cloud environment and develop the ability to proactively react to adverse cyber incidents. Diagram below illustrates how Secure Software Development Lifecycle (SSDL) model can integrate security capabilities at each step.



Design and develop

Security guardrails should be enforced during every step of the design and develop phase, so that any misconfigurations are fixed earlier in the software lifecycle in order to enable secure infrastructure provisioning.

Specific aspects to embed in this phase include:

- Policy-as-code guardrails should be codified with industry-leading practices, Deloitte secure guardrails, as well as any requirements stipulated by compliance, cloud governance and custom requirements.
- Scanning for misconfigurations within default and custom-created policies should be integrated with developer tools such as IDE, and VCS and CI/CD.
- Secrets identification implementation, including passwords and tokens in IaC templates in IDEs, command-line interfaces (CLIs), local or in-registry container images, pre-commit, and CI/CD tooling, etc.
- Automated feedback and code fixes for identified misconfigurations should be in place.
- Feedback from tools such as continuous compliance monitoring, threat monitoring, and control improvement's feedback should be implemented in policy as code in regular iterations. They should be used to scan the code files or images for misconfigurations at every stage of development.

- Security baselines should be revisited frequently in order to fine-tune existing guardrails used for scanning.
- A Zero Trust framework should be enabled throughout design and development. Scanning of architectures with advanced threat modeling use cases should also be implemented.

Build, test, and deploy

Once dependencies and code files are packaged together, the entire build should undergo a more in-depth analysis of security threats and misconfigurations in the application or code.

Specific aspects to consider during this phase include:

- The integrated scanning of IaC templates, container images, and serverless functions with CI/CD tools. The goal here is to identify vulnerabilities in any code files, operating systems, and open-source packages built into container image layers or serverless functions.
- Detection and prevention of misconfigured code builds found in the repository. Builds will not achieve their expected value unless the misconfigurations identified are not fixed.
- Added guardrails that block images with less-than-severe vulnerabilities before they are pushed to production.
- The return of automated pull requests with detailed remediations to the source code location for identified misconfigurations.
- Performance of vulnerability scans to harden images. This is to leverage build-time scanning and trusted registries for a more secure container image supply chain.
- Putting alert notification and real-time dashboards in place for monitoring of important events during deployment activities.
- Incorporation of access controls to in order to segregate duties, along with authorization tollgates to review and accept build requests.
- Frequent revisiting of security baselines in order to fine-tune existing scanning guardrails.

Monitor and operate

The code released into the environment needs to be normally monitored to identify security or configuration drifts.

Specific aspects to be considered during this phase include:

- Continuous monitoring of infrastructure configurations for compliance, technical, or baseline control drifts.
- Auto correction of drifts in near real time through baseline configurations.
- Continuous monitoring of vulnerabilities across host operating systems (OS), container images, and serverless functions.
- Implementation of runtime defense on hosts, along with the establishment of a regular patch management cadence.
- Establishment of a single unified console provide continuous visibility across all deployed assets and their posture or configurations.
- Establishment of user entity behavior analytics (UEBA) to monitor cloud environments for unusual user activities in order to discover insider threats and potential account compromises.
- Continuously monitoring of unused or risky permissions tied to cloud entitlements to remove unwanted access to cloud resources. This is done by detecting overly permissive or unused access policies automatically and evaluating affected permissions that should exist.
- Implementation of orchestration playbooks and automation workflows. The goal here is to notify the appropriate stakeholders of important security events and to establish change management.
- Continuous alignment with industry standards and leading practices. This is accomplished through routine management and fine-tuning of rules governing cloud-native infrastructure, docker configurations, containers, images, nodes, plugins, and services to determine a secure environment.

Deloitte and Palo Alto Networks Accelerators for SSDL

Deloitte Accelerators
Deloitte solutions leveraged to achieve the SSDL model:

- Security automation that provides for the implementation of security automation use cases and patterns to enable near-real-time remediation of security drifts—from baseline configuration standards to secure core cloud services.
- Cloud controls framework that provides a library of policies mapped to industry-leading standards. It can be leveraged to fine-tune organizational security guardrails.

Palo Alto Networks Solutions
Palo Alto Networks solutions integrated and referenced as part of the SSDL model:

- With scanning support for IaC templates, container images, open source packages and delivery pipelines, Prisma Cloud provides code security backed by an open source community and years of expertise and threat research. With connected visibility and policy controls, engineering teams can secure their full stack without leaving their tools, while security teams can ensure that all deployed code is secure.
- Prisma Cloud's unique Cloud Security Posture Management (CSPM) solution eliminates cloud blind spots, proactively address risks and reduces the complexity of securing multi-cloud environments while radically simplifying compliance.
- The Cloud Workload Protection module for Prisma Cloud delivers flexible protection to secure cloud VMs, containers and Kubernetes apps, serverless functions and containerized offerings like Fargate tasks. With Prisma Cloud, DevOps and cloud infrastructure teams can adopt the architecture that fits their needs without worrying about security keeping pace with release cycles or protecting a variety of tech stacks.
- The Cloud Infrastructure Entitlement Management module for Prisma Cloud addresses the complexities of entitlement management across multi-cloud environments by providing deep visibility and control of permissions, and automatic remediation of overly permissive and other risky permissions.
- Security automation and orchestration that provides automation-driven detection, investigation, and response workflows for security operations.

The strength of the Deloitte and Palo Alto Networks relationship

Deloitte's award-winning Cyber & Strategic Risk consultants have joined forces with Palo Alto Networks and its security capabilities platform. Together, we're working to provide a broad range of solutions that simplify the complex software development lifecycle, while increasing speed, agility, and enablement so that organizations like yours can better protect their infrastructure and workloads at every stage of the development lifecycle.

Our joint solution can help you create a cyber-minded culture for your organization so that it can move forward fast and stronger, fuel more innovation, and stay more resilient in the face of persistent and ever-changing cyberthreats—all while accelerating time to market and reducing costs.

Authors

PALO ALTO NETWORKS ALLIANCE LEADERS
Kieran Norton
Principal
US Cyber & Strategic Risk
Deloitte & Touche LLP
kinorton@deloitte.com

PALO ALTO NETWORKS ALLIANCE LEADERS
Jane Chung
Managing Director
US Cyber & Strategic Risk
Deloitte & Touche LLP
jachung@deloitte.com

Siddharth Kantroo
Advisory Senior Manager
US Cyber & Strategic Risk
Deloitte & Touche LLP
skantroo@deloitte.com

Anthony Polzine
Senior Manager
Global Partner Solution Architect
Palo Alto Networks
apolzine@paloaltonetworks.com

Deloitte.