SANS

Analyst Program

Whitepaper

Building Security from the Ground up with Secure by Design

Written by Eric Johnson, Bertram Dorn, and Paul Vixie

aws

Introduction

System design often prioritizes performance, functionality, and user experience over security. This approach yields vulnerabilities that can be exploited in the product and across the supply chain. Achieving a better outcome requires a significant shift toward integrating security measures into every stage of development, from inception through deployment.

As the threat landscape continues to evolve, the concept of Secure by Design (SbD) is gaining importance in the effort to mitigate vulnerabilities early, minimize risks, and recognize security as a core business requirement. SbD aims to reduce the burden of cybersecurity and break the cycle of constantly creating and applying updates by developing products that are foundationally secure.

A total of 26,447 critical vulnerabilities were disclosed in 2023, surpassing the previous year by more than 1,500.[1]

Insecure design is ranked as the number four critical web application security concern on the Open Web Application Security Project (OWASP) Top 10.[2]

Supply chain vulnerabilities are ranked fifth on the OWASP Top 10 for Large Language Model (LLM) Applications.[3]

The Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), Federal Bureau of Investigation (FBI), and international partners including the Five Eyes (FVEY) intelligence alliance have adopted the SbD mindset and are evangelizing it to help encourage proactive security practices and avoid creating target-rich environments for threat actors.

More than 60 technology companies—including AWS, Microsoft, and Google—recently signed CISA's Secure by Design Pledge as part of a push to put security first when designing products and services.[4]

This chapter explores what SbD actually means and discusses its benefits, cultural aspects, key considerations, and action items that can set you on the path to successfully embedding SbD into your security strategy.

Understanding Secure by Design and Secure by Default

The term Secure by Design is often confused with Secure by Default. These are two distinct but complementary elements of a holistic security strategy.

- **Secure by Default** is a user-centric approach that indicates the default settings of a product are secure out-of-the-box and resilient against common exploitation techniques, without the need for additional security configuration.

---

1 "2023 Threat Landscape Year in Review: If Everything Is Critical, Nothing Is," January 2024, https://blog.qualys.com/vulnerabilities-threat-research/2023/12/19/2023-threat-landscape-year-in-review-part-one

2 "OWASP Top Ten," https://owasp.org/www-project-top-ten/

3 "OWASP Top 10 for Large Language Model Applications," https://owasp.org/www-project-top-10-for-large-language-model-applications/

4 "Secure by Design Pledge," www.cisa.gov/securebydesign/pledge

- **Secure by Design** is a developer-centric approach that goes beyond implementing standard security measures to evaluate and address risks and vulnerabilities at every stage of the development life cycle—from design to deployment and maintenance—rather than reacting to them later.

Both ensure that security is inherent. Together, they work to establish a solid foundation for proactive security, build trust with customers, and increase the level of difficulty for threat actors seeking to exploit products and systems.

Secure by Design offers more flexibility to help protect resources and withstand threats that originate outside of architectural components. It allows you to use products with different options and settings, so the outcome aligns with your risk tolerance level.

With SbD, the security of architectural components that products are built around cannot be altered without changing their fundamental design or setup. SbD principles can be applied to components ranging from IT workloads to services, microservices, libraries, and beyond.

Another way to think of SbD is to consider the topology of a space, such as a house. An SbD setup should have only closed, finite rooms in the configuration space (house) that do not allow access to an infinite space (outside of the house) except through well-defined and carefully controlled ingress and egress points. This absence of configuration space options facilitates added security. If you don't make design principles accessible to builders, then they're creating IT workloads in a secure environment.

When software is in the cloud, SbD helps eliminate access points. Identity and access management (IAM) is your first line of defense, as IAM misconfigurations can lead to misconfigurations and unsecure usage elsewhere. An example of an SbD approach in an IAM system for distinct principals (IAM users, federated users, IAM roles, or applications) is to rely on testable outcomes that make them atomic. Because IAM is inherently based on the "default deny" principle that either explicitly allows or implicitly denies access, SbD helps you lay the foundation of a secure IAM setup for builders and operators within the cloud environment as part of an overarching, centralized IAM system that is accompanied by centralized logging. New design elements should automatically inherit the secure setup; otherwise, they shouldn't work.

Embedding Secure by Design into Your Security Strategy

Incorporating SbD into your overall security strategy can help your organization minimize potential risks, boost productivity, build trust with customers and partners, and reduce costs over time by developing products and services that require less patching after delivery. Software development life cycle (SDLC) processes, automation, defense-in-depth, artificial intelligence (AI), threat modeling, and compliance are key factors to keep in mind.

Integrating SbD into the SDLC

SbD contrasts with more traditional development approaches that introduce security measures as additional layers to the end product. Shift left and DevSecOps[5] are related concepts for incorporating security throughout the SDLC that result from an SbD approach.

Commonly used SDLC models such as waterfall, spiral, and agile don't address software security in detail, so secure development practices need to be added to set foundational security. Additionally, in a cloud environment, infrastructure is also code that should fall under the purview of the SDLC.

The National Institute of Standards and Technology (NIST) Secure Software Development Framework (SSDF), also known as SP 800-218, can support efforts to strengthen the security of your SDLC. The SSDF describes a set of high-level practices based on established standards, guidance, and secure software development practice documents from organizations such as SAFECode, BSA, and OWASP. The framework is divided into four groups (see Figure 1) that are designed to help prepare the organization's people, processes, and technology to perform secure software development, protect software from tampering and unauthorized access, produce well-secured software with minimal security vulnerabilities in its releases, and respond to residual vulnerabilities. Although it's not a checklist—and the degree to which you choose to implement the practices depends on your organization's requirements—it can help you adopt well-understood best practices and ensure team members across all phases of the development pipeline assume responsibility for security.

Supporting SbD with Automation

Two areas for automation in relation to SbD workloads are important in the effort to maintain healthy and secure setups. The first is preventive controls that ensure configurations can be rolled out only in a secure mode that is defined by the design.

Continuous integration and continuous delivery (CI/CD) pipelines that help automate the software delivery process are substantial contributors to SbD environments, as they include a comprehensive set of checks to be run—such as firewall settings, OS configurations, libraries used, security-related reviews, and software components used—before a target configuration is implemented.

The second area of automation includes detective systems, which can identify noncompliant components or configurations. Misconfigurations generally shouldn't happen within SbD setups, as they are largely prevented through the design and preventive controls in the implementation. Additionally, it is important to note that a vulnerability in a system due to either the design or the implementation may not pose an immediate problem, if the design includes defense-in-depth elements that protect the overall system despite any individual flaws. Nevertheless, if a detective system finds something that doesn't adhere to the design, it's a signal that the design needs to be

---

[5] "What is DevSecOps?" https://aws.amazon.com/what-is/devsecops/

improved and/or preventive controls need to be added, and that the anticipated "closed" space has not, in fact, been closed.

Recognizing the Importance of Defense-in-Depth

In an SbD approach, it's important to recognize that no matter how careful the design or implementation, systems or controls may fail. Leveraging diverse security measures, such as network hardening and security system integration, along with secure software design, can help you address threats and eliminate single points of compromise, so the failure of one control doesn't lead to the failure of the overall protection provided.

Figure 1. Establishing Secure Development Practices

If a builder successfully places a potential target, such as sensitive data or critical workloads, in an SbD structure, that target is protected by a layered defense. A secure design can, and should, be embedded (or nested) into another secure design to form a shield of defensive layers that support each other. In situations involving an unsecured resource, such as a legacy application with undefined software supply chain risks, the resource can be protected with an SbD approach by creating a closed space around it.

If the design needs to be open to enable some business processes, a layered defense can address threats and either prevent a breach or limit potential damage.

Applying SbD to Artificial intelligence (AI)

The need for SbD applies to AI like any other software system. Organizations in all industries have started building generative AI (GenAI) applications using large language models (LLMs) and other foundation models (FMs) to enhance customer experiences, transform operations, improve employee productivity, and create new revenue channels. As you explore the advantages of GenAI, it's important not to let innovation take precedence over security.

82% of business leaders view secure and trustworthy AI as essential for their operations, but only 24% are actively securing GenAI models and embedding security processes in AI development.[6]

FMs and the applications built around them are often used with highly sensitive business data, such as personal data, compliance data, operational data, and financial information, to optimize the model's output. Risks can stem from the setup around training data, the origin and nature of training data, prompt design, and the use of techniques that can lead to hallucinations, all of which

[6] "Securing generative AI: What matters now," May 2024, www.ibm.com/downloads/cas/2L73BYB4?mod=djemCybersecruityPro&tpl=cs

can impact the usability of results. To protect users and data, security needs to be built into machine learning (ML) and AI with an SbD approach that considers them to be part of a larger software system and weaves security into the AI pipeline. A "kill" switch may be needed at the output of a GenAI system to prevent it from leading in unwanted or misleading directions. Building finite, closed spaces can help you secure training data against poisoning and misuse. The finite space for training data should be empty up front, and individual decisions about introducing datasets for the targeted model should be documented. To further address risks, additional measures should be taken to keep them SbD. These may include placing output in a closed space and, in security-relevant situations, specifying that it should only leave the space for use if there's a plausibility, applicability, and sanity check performed by humans in the loop.

Integrating an AI/ML bill of materials (AI/ML BOM) and cryptography bill of materials (CBOM) into BOM processes can help you catalog security-relevant information, and gain visibility into model components and data sources. Additionally, frameworks and standards such as the NIST AI Risk Management Framework (AI RMF 1.0), the HITRUST AI Assurance Program, and ISO/IEC 42001 can facilitate the incorporation of trustworthiness considerations into the design, development, and use of AI systems.

Identifying Threats in the Design Phase with Threat Modeling

Threat modeling is an essential part of an SbD approach that can help you analyze the security of a product from an adversarial perspective, identify potential threats, and determine responses to these threats.

Threat modeling in the design phase fosters a culture of secure design and implementation, which in today's landscape includes infrastructure, configuration management, and application code. Threat modeling exercises conducted during design allow development and security teams to conceptualize and then document potential threats before code has been written, and can save time and money by avoiding rework or escalations later in the development process. Threat models should be treated as living documents that can be integrated into your SDLC processes and evolve with experience and learnings, as well as the overall product evolution and threat landscape over time.

During a threat modeling exercise, mitigation activity should focus on both the design and the available technology.

CVE-2024-3094—a critical supply-chain compromise recently found in the XZ Utils data compression library—can be used as an example. The malicious code leverages testing mechanisms in the build process and attempts to weaken the authentication of secure shell protocol (SSH) sessions via SSHD (the server-side daemon process for SSH) within some operating system environments to allow unauthorized access. The accessibility of operating systems should be considered when threat modeling. The challenges of the library supply chain are included in a standard lift-and-shift cloud migration approach. If the access approach to affected XZ Utils versions 5.6.0 and 5.6.1 isn't limited to one daemon (in combination with the operating system's login process), and includes

your cloud provider's IAM security layer, the pure vulnerability of the SSHD would be mitigated. The finite space of the IAM setup would create finite space for the login, which aligns with an SbD approach.

Rethinking threats to the login process itself can lead to either network-based control of the communication as discussed, or to the login process being embedded in a different environment, which helps to prevent the threat. The cloud- and IAM-based login design can also provide you with benefits through the scalability or responsibility shift that comes with the cloud. Keeping the IAM system secure is a critical task for the cloud service provider (CSP) and part of its core business functions. The centralized logging and monitoring capabilities provided by the CSP's IAM system can help you ensure that only authorized users have access to sensitive data and resources.

Design-focused exercises should be prioritized and conducted for common threats starting with physical aspects, and moving up to the network layer, operating system and setup topics, software topics, database queries, storage usage, authorization and authentication methodologies, and deep into software and hardware design supply chains. Rather than using the most flexible design with fine-grained configuration options, threat modeling can help you define the most secure design. If you need customization, you can document adjustments and run them through an approval workflow. Misconfigurations can be mitigated using a combination of policy guardrails and detective automation.

The Impact of SbD Approaches on Threat Modeling

To stay with the model of spaces, an SbD approach closes the topological space and creates a finite environment, which has an impact on typical threat-modeling trees. Imagine a bowl or torus (see Figure 2) that represents all interaction possibilities (instead of a tree structure with open ends on all sides). As all possibilities in a closed space are known, they can be designed to be safe by nature. The impact of this approach is that—because the resources within the space are safe—the actors inside can navigate freely within those constraints. The space needs to be reviewed periodically to determine whether the implementation of this freedom based on limited choices meets your business requirements. It should be noted that open and closed spaces are not to be directly equated with network boundaries. Network boundaries can be helpful to design and defense-in-depth, but zero trust principles mean that network boundaries are never fundamental to a secure design.

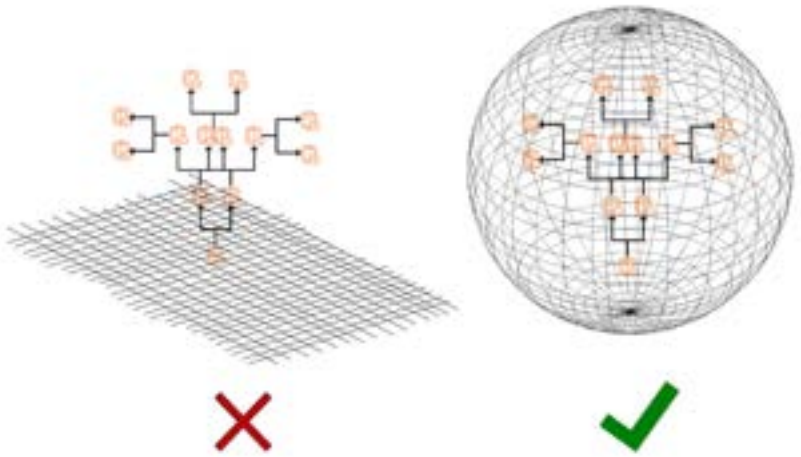

Figure 2. Closing Topological Space with an SbD Approach

This allows you to scale out new deployments without reiterating on the security setup, including access control, logging, configuration recording, connectivity, restore possibilities, and all the other security domains that are required after conducting threat modeling and compliance analytics.

The SbD nature of the created space (through automation or systems on the borders) keeps new resources of the same type in the safe state and free from inherited threats. Resources with unwanted configurations, states, or dependencies cannot be deployed. However, the design's assumptions should be challenged as often as your commercial requirements allow. Each vulnerability scan with findings should be tested for an impact on design aspects. New business requirements and compliance considerations also may necessitate changes or add to your design targets.

Guiding Principles

There are three guiding security design principles to consider when applying SbD to threat modeling:

1. Address threats through design first before turning to a technical or organizational measure to try to eliminate the threat area. Choose design options that help keep the threat out of your product, so you don't need to address them later.
2. Monitor the logical borders created by closed spaces to gather baselines. This helps create enforcement automation through filter and approval technology. Typical examples of borders include firewalls, web application firewalls (WAFs), landing zones, system-call jails for preventing unwanted interaction with the operating system (such as creating users), IAM settings, closed-source libraries, and software repositories.
3. Design your spaces to allow changes to flow within. You can achieve this by defining behavior outcome at the design level and having your builders make decisions along this path in the design space you create.

Leveraging SbD to Simplify Compliance

An SbD approach isn't a direct path to legal compliance. However, the use of SbD practices can help you conform to internal technical compliance requirements as well as regulatory requirements that include technical and organizational measures (TOMs) addressing data management processes in relation to the General Data Protection Regulation (GDPR) and detailed guidance from the Payment Card Industry Data Security Standard (PCI DSS). SbD can also help you adhere to voluntary best practice frameworks, such as the International Organization for Standardization (ISO) 27000-series information security standards and the NIST SSDF.

An SbD approach can help you meet requirements across people, process, and technology. From a people perspective, you can provide mandatory, role-specific training on secure coding best practices. From a technology perspective, your design can be set

to automatically initiate and enforce backups or allow only preconfigured, finite, and known network connections. From a process perspective, you could build a deployment pipeline with an attached ticket system that can automatically enforce necessary processes and documentation.

You can get there by defining your compliance requirements from the domains to the controls, and then iteratively working backward to meet them. Instead of linking the controls directly to TOMs, such as architectural components, configurations, processes, and procedures, the goal is to link them to design principles, which are then implemented in TOMs *outside* of the target system. The TOMs are then part of the surrounding design and unchangeable configurations, rather than part of the workload itself.

Consider zero trust, for example. If your target system is in an environment that enables communication only after context-based authentication and authorization verify the identity of the actor attempting access, you can meet a set of technical compliance requirements around access and user management and facilitate a finite and closed space by allowing only approved connections.

Planning for the Short- and Long-Term

SbD environments address vulnerabilities early with a focus on building closed spaces that include guardrails for workloads. This creates important short- and long-term effects to consider.

Short-Term Effects to Mitigate

In the short term, builders who aren't accustomed to keeping security in mind at every stage of development may feel their agility is being constrained. Developers are often encouraged to focus on launching new products as quickly as possible to support competitive advantage. An SbD approach may not allow them to use all the code libraries and system access they want in the process, which can impact time to market.

On the other hand, supporting builders with a design phase in which the main security and technical compliance aspects are handled can make them feel safe, because the design helps prevent insecure configurations. This is typically the situation in cloud systems, where builders only get access to approved cloud services over approved authorization and authentication paths, with prepared logging and reporting (landing zones). The what, who, and where is defined and controlled by preventive automation, such as a centrally managed IAM system and infrastructure as code (IaC).

Landing zones with built-in policy guardrails may initially add friction to the developer workflow and decrease agility during a period of adaptation. However, the freedom provided within well-defined constraints will ultimately pay off in terms of both better security outcomes and the agility needed to help you achieve business goals.

Some may wonder: Is speed of the essence for the product, or should you invest more in its security? The answer provided by experience is increasingly clear. Organizations that prioritize speed of execution over security are paying a heavy price. A properly designed SbD engineering framework provides more than adequate speed and agility. Not only will your products—and your customers—be better protected with security incorporated into every development decision, but your security efforts also will ultimately cost less overall. In many cases, *good* security can't simply be added after the product is substantially complete.

Another short-term aspect of the targeted closed space of an SbD setup to consider is how to deal with new technology, testing, and exploration. Testing is critical to verify the quality and reliability of applications. This requires teams to create test suites and mock data to maximize code coverage in lower-level environments. Doing so creates a secure space for development and testing, which is often managed by version control systems and continuous delivery pipelines.

Long-Term Effects to Consider

In the long term, designs can ultimately create challenges if they are too fixed and can't allow for changes in technology. You should, therefore, check design decisions for unnecessary rigidity to prevent negative effects.

The evolution of cryptography is one example. If a design specifies a specific algorithm or key length, its efficacy will slowly erode. It could either stop providing the security needed to protect sensitive data due to increases in computing power—as happened with the Data Encryption Standard (DES), which was withdrawn and superseded by Advanced Encryption Standard (AES)—or it could cause a device to stop working because of an expired certificate that it cannot renew.

To make products and services more sustainable, the SbD approach requires a balance between the usability of the product, fundamental parameters, and malleability of the design. Designs should take long-term effects into consideration by calling out functions and their outcome and risk surface but staying out of technical details.

Facilitating a Culture of Security

An SbD approach facilitates cultural transformation that makes security a shared responsibility for everyone involved in development. It applies to different functions involved with your IT workloads on the path from identifying requirements to creating a design defining the output to technical and organizational measures that are then looped back through controls.

The Builder

Builders are the main target of an SbD approach. Although they may initially have concerns over access and resource limitations—as well as debugging and implementation challenges if a design does not offer the anticipated levels of technical readiness—builders also will realize benefits.

The design should provide builders with clear guidance and eliminate risk decisions that can be made without undergoing a security approval process. Because builders may not always realize when they are making a risk-based decision during the development process, the design should prevent them from having to make them in the first place.

Additionally, builders can sometimes address challenges with the borders of the created space through the design. For example, it may be easier to design an intake process for code libraries and other elements of the supply chain to get software components from existing outside repositories rather than create new components that may also introduce new risks. It's important to note that this intake process would be subject to threat modeling as part of your SbD approach.

The Owner

Business process owners define the target behavior and, therefore, the design space. In many cases, business requirements are found to be broader than expected. Regulatory considerations, commercial aspects ranging from costs to time to market, and long-term strategies for asset development should be incorporated into the design so that its parameters can be modeled as input data for technical integration.

Business and IT workload owners are also key players in the process of working backward from the most secure design to a desired working point that balances security needs with commercial aspects. Informed decisions should be made, with clear documentation and defined risk takers.

The Supervisors

Supervisory functions, such as design review boards or auditors that define and potentially manage the data exchange between systems, address the integration feasibility of the design through technical or organizational measures. They implement preventive controls to set the borders of the finite space defined by the design and can use detective controls to guide the design into the future. Additionally, the automated implementation of those controls, through configuration management systems, can generate evidence and documentation to prove the compliance status of the overall IT workload. In cloud environments, where all configurations are known, the automation of evidence generation is particularly important. Because these controls are running on the principal of SbD to create finite configuration spaces, the desired result is a technically compliant state.

The Insurance

When accompanied by a layered defense for situations in which a workload needs to deal with infinite configuration spaces, the SbD approach shifts the risk discussion away from individual technical decisions to the design layer. However, the insurance aspects of transferring or sharing risks also should be considered. Managed services, especially those with higher integration layers, offer the ability to transfer risk and reduce your operational burden. The shared responsibility between your organization and the managed service provider (or CSP) encapsulates risk, and auditor-issued reports, certifications, accreditations, and other third-party attestations provide visibility into the effectiveness of their security and compliance posture. Partnering with providers can help you cut off areas of risk and minimize the potential impact of security incidents as part of an SbD approach.

Benefits

A robust SbD approach establishes a solid foundation that reduces risks and yields security benefits for your development teams—and your business.

Scalability

Operations inside an SbD setup allow you to quickly scale, without reiterating security settings. This is particularly beneficial in environments where the demand cannot be predicted precisely up front. An SbD approach helps create well-architected landing zones that are both scalable and secure. Automation through code pipelines, including automatic code checks against attempts to open the room, are a key element here. These pipelines also add to detective controls to help identify attempts (or the need) to cross the borders of the design. Systems that execute these pipelines concentrate risk through the need to be able to execute everything that is required. Therefore, they should be outside of the target design in their own closed SbD room. This provides a starting point your organization can use to efficiently launch and deploy workloads and applications with confidence in your security and infrastructure environment.

Repeatability

Having prepared spaces also allows you to repeat setups in an agile way. With an SbD approach, you can build products and services that are designed to be foundationally secure using a repeatable mechanism (see Figure 3) that can anchor your development life cycle.

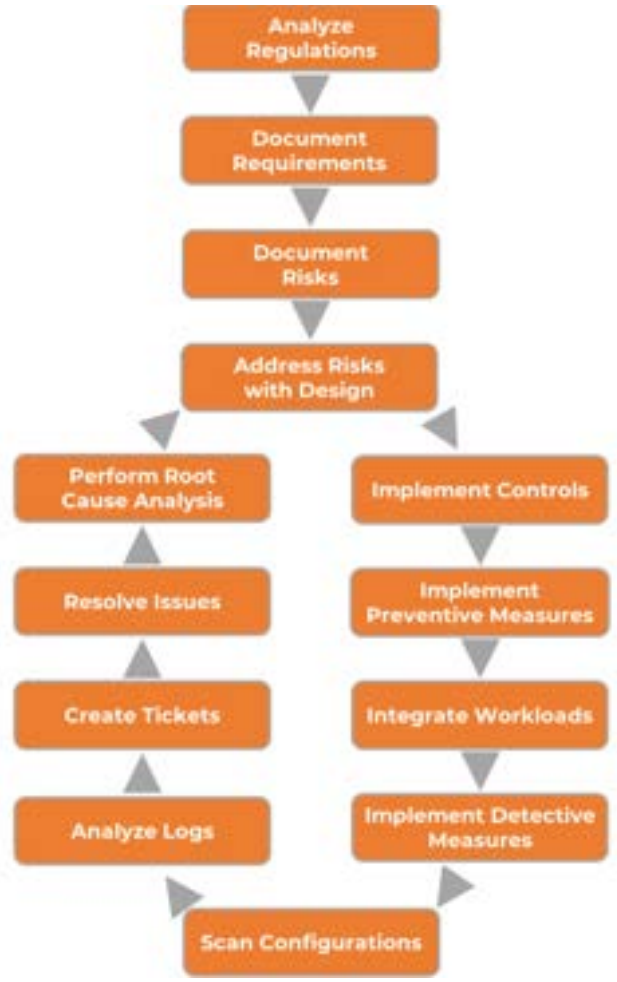

Figure 3. SbD as the Anchor of Your Development Life Cycle

Agility

While your builders may be concerned about the access and resource limitations associated with an SbD approach, agility inside a closed space can be higher in the long term. When the design of an environment is secure, builders inside the SbD configuration do not need to rethink the security setup and can concentrate on their areas of expertise. By weaving security into your development practices, your organization can become more agile, resilient, and responsive to threats.

Sustainability

A solid SbD approach includes built-in feedback loops through detective controls that facilitate sustainability by enabling you to analyze data and leverage insights to enhance the security of your products, services, or processes. If the design considers future developments in the technology—such as cryptography changes, for example—following them should be possible by design. This leads to products and services with a longer lifetime, with potentially fewer changes and iterations, and a stable interaction surface.

In a cloud environment, log data can be used to create detective controls. Detection of anomalies, and failed attempts to configure things outside the closed space should lead to documentation (through tickets) and can drive the creation of new controls that can further advance detection. This creates a flywheel effect that can help you tighten security and cut off open treats. The closed space remains closed while taking care of the dynamics on its borders.

Manageability

Manageability functions such as logging, reporting, and gathering data for compliance purposes can generally be built into the design and don't need to be rethought. Included preventive controls will automatically generate the data needed to keep the IT workload under control. A predesigned operating setup for compute instances, for example, can include backup and restore, logging, access management, patch management, inventory management, and telemetry data functions that are automatically rolled out. Nowadays, you can orchestrate these things through automated systems and document them with detective controls.

Key Considerations on the Secure by Design Path

Security is not like a Boolean parameter that's either true or false. It is closer to a quantum state, since known security posture is constantly evolving. Design changes and implementation efforts should be key considerations in your SbD approach.

Design Changes

Threats to new and existing technologies are constantly evolving. SbD preventive controls, such as firewalls and runtime security agents, can help security teams respond quickly to an intrusion. Consider a scenario in which your security team has discovered

a threat actor compromising the environment through a vulnerable library. Following an agile workflow, the security team quickly writes a firewall policy and deploys the rule through a CI/CD pipeline to thwart the intrusion. From there, additional root cause analysis can identify permanent design improvements to help prevent future incidents. Security teams performing continuous, incremental design enhancements will be in a better position to respond quickly to evolving threats.

Implementation Efforts

The path to an SbD approach includes an additional step between requirements and practical implementation. You will need time to qualify design-level mitigations, and traditional conversations and documentation may be required to establish the "why" and "how" behind your approach.

In addition to setting aside time for the design phase, focus on standardizing approaches that can be reused by others. You also should choose your tech stacks carefully, keeping an eye on complexity and related security overhead. Consider leveraging cloud services to address problems, instead of building everything on your own.

The tool landscape also should be approached through an SbD lens. Free selection of tools, such as programming languages, and methodologies in software development can create more dependencies (and open spaces) than the risk appetite for the deployment allows. Time to market and implementation considerations might outweigh security concerns related to the selected language or programming framework with its dependencies, which is exactly what you should consider with care.

Using services with a higher integration level, such as cloud services, can reduce your implementation efforts. Critical capabilities, such as IAM and connectivity, are already designed and have security built in, which helps provide you with a closed risk environment by design.

Getting Started

Taking a new approach to security can be daunting, especially for organizations used to focusing on "check the box" compliance exercises. Five key action items that can help you avoid frustration and set you on the path to successfully implementing SbD include:

- **Identify your core SbD pillars—**Evaluate a matrix of your technical domains with the security domains that are called out by your business processes. Technical domain examples include logging and security domain integrity while design examples include mandatory checksums and authenticated encryption. Each node in the matrix will require a decision to be made regarding how to address the associated security domains.

- **Define the scope of your design—**Attempt to eliminate risks within the matrix with a design change. If elimination is possible, consider the potential effects on the business and on TOMs. Document and communicate the elimination process to relevant stakeholders.
- **Validate technical feasibility—**Verify whether the identified elimination process can be implemented with your technical or organizational capabilities. There may be commercial aspects that prevent the most secure design. If there are conflicts and/or a need to move away from the secure design to a less secure setup, verify that the change and its outcome are understood. Document economical tradeoffs and undertake a risk-acceptance process with the owner of the workloads and the builders.
- **Stay flexible—**Emphasize the need to remain flexible with all your SbD stakeholders. Known and established architectural components may not meet actual design options, which presents an opportunity to invent and simplify. Keep the design thinking flexible, as well. Technical needs might create topics that require urgent attention. In such a case, the design is temporarily secondary to security needs. However, the design should take the lead in incorporating learnings from the emergency into the overall setup and components.
- **Review your design continuously—**Implement detective and preventive controls and pay attention to their findings. Expect to discover vulnerabilities, and continuously feed them into your design processes. Allow your products and services to react to design changes and avoid one-way-door design decisions that could have significant and irrevocable consequences.
- **Create open lines of communication—**Set up processes to gather feedback from stakeholders and users about security issues with regular meetings to update your leadership team. Measuring progress is key to quantifying positive impact. However, much of the impact of an SbD approach amounts to measuring what didn't happen or what might otherwise happen if security doesn't stay central to your efforts. Although it's impossible to measure these outcomes with certainty, you can present metrics that convey a reasonable view of progress. Sample metrics that can help you account for both direct costs, such as patch management, and indirect costs, such as brand reputation, might include:
    - How much your organization spends each year fixing security issues after software is deployed
    - How much you could or have saved by building security in at the design phase
    - Changes in customer satisfaction since developing products that require fewer patches

> **Keeping the world safe as the digital economy grows is a big challenge that can only be accomplished through automation. Secure by Design (SbD) is how that automation vitally appears at the system design level. Every digital architect needs to know what SbD is and how to apply it. —Paul Vixie, AWS Deputy CISO, VP, and Distinguished Engineer**

Conclusion

Good security is the key to experimenting with new technologies. A proactive, Secure by Design approach to development that builds security from the ground up allows you to identify and fix vulnerabilities early, increase cost efficiency, and create more resilient products. The use of closed, finite spaces for development activities can provide your builders with a secure environment to work in, and help you withstand threats that originate outside of architectural components. As you consider your organization's development processes, start thinking about the spaces team members are using for engineering activities. Are they open or closed? Taking a new approach may seem daunting, but establishing the right foundations and keeping key considerations in mind along the way can help you successfully embed SbD into your security strategy—and build trust as you innovate.

Sponsor

SANS would like to thank this paper's sponsor:

BUILDING SECURITY FROM THE GROUND UP WITH SECURE BY DESIGN

PAUL VIXIE

ERIC JOHNSON

CISA → PLEDGE ENCOURAGES PROACTIVE SECURITY BEST PRACTICES

SECURE BY DESIGN

ADDRESSES RISKS & VULNERABILITIES AT EVERY STAGE OF THE SOFTWARE DEVELOPMENT LIFECYCLE

INCLUDES BEING SECURE BY DEFAULT

THE MOST SECURE IS IN A BOX, TURNED OFF & STASHED AWAY

WOULD I RATHER INVEST UP FRONT ? OR MAJORLY LATER

STRATEGY

NIST SP800-218 SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF)

FORMAL RULES WRITTEN DOWN & HAVE A PLAN TO AUDIT

CI/CD AUTOMATION TO VALIDATE SECURITY POLICIES

SECURITY IS JOB ZERO

CONSIDERATIONS

INCREMENTAL DESIGN

TESTING IS CRITICAL

FIGURING OUT IF YOU BOUGHT THE RIGHT PRODUCT

GROWING TOO FAST

SECURITY CHAMPIONS

YOU HAVE TO BE A GOOD OWNER & STEWARD

THREAT MODELING

WHO IS GOING TO UNDERSTAND THIS BEST?

ME OR THREAT ACTOR

EVOLUTION

EVERYTHING IS CONSTANTLY IN MOTION

WE NEED AUTOMATION TO HELP US

STARTS AT THE TOP

CULTURE OF SECURITY

EVERYONE CARES ABOUT IT AND IS EVERYONE'S JOB

SECURITY IS TAKEN SERIOUSLY

- BUILDERS
- OWNERS
- SUPERVISORS
- INSURANCE

BENEFITS

- SCALABILITY
- MANAGEABILITY
- AGILITY
- REPEATABILITY
- SUSTAINABILITY

@SANSINSTITUTE #CLOUDSECURITY

CLOUD SECURITY EXCHANGE 2024

VIRTUAL EVENT

Tuesday August 27 2024 | 11am ET

aws Google Cloud Microsoft

@mindseyecreative MIND'S EYE CREATIVE