# Akraino Security Development Lifecycle



## Overview

### Akraino SDL Roles

Security advisor/Privacy advisor

- Auditor
- Expert

Team security champion/privacy champion

- Negotiate, accept, and track of minimum security and privacy requirements
- Maintain clear lines of communication with advisors and decision makers

## Security training

- Secure design
- Threat modeling
- Secure coding
- Security testing
- Privacy
- Security response processes

## Requirements

## Verify if the project is subject to SDL policies

- Deployed in a business or enterprise environment
- Processes personally identifiable information (PII) or other sensitive information
- Communicates regularly over the Internet or other network

## Security bug reporting tools

Security bug effects

- Not a Security Bug
- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

Security bug cause

- Not a security bug
- Buffer overflow/underflow
- Arithmetic error (for example, integer overflow)
- SQL/Script injection
- Directory traversal
- Race condition
- Cross-site scripting
- Cryptographic weakness
- Weak authentication
- Weak authorization/Inappropriate permission or access control list (ACL)
- Ineffective secret hiding
- Unlimited resource consumption (Denial of Service [DoS])
- Incorrect/No error messages
- Incorrect/No pathname canonicalization
- Other

## Security bug bar

## 3rd-party code licensing security requirements

## Security Plan

- Team training
- Threat modeling
- Security push
- Final security review

## Cost Analysis

- Security risk assessment
  - What portions of the project will require threat models before release.
  - What portions of the project will require security design reviews before release.
  - What portions of the project will require penetration testing (pen testing)
  - Any additional testing or analysis requirements the security advisor deems necessary to mitigate security risks.
  - Clarification of the specific scope of fuzz testing requirements
- Project privacy impact rating
  - P1 high privacy risk
  - P2 Moderate privacy risk
  - P3 Low privacy risk

# Design

## Risk Analysis

### STRIDE threat model analysis

STRIDE:

- **S**poofing of user identity
- **T**ampering
- **R**epudiation
- **I**nformation disclosure (privacy breach or data leak)
- **D**enial of service
- **E**levation of privilege

Threats and vulnerabilities
External code
Threat models
Design review for P1 privacy projects
Detail privacy analysis

### NEAT security user experience

- **N**ecessary
- **E**xplained
- **A**ctionable
- **T**ested

## Best Practices

### Secure design principles

•Secure defaults
•Defense-in-depth
•Separation of privilege
•Least privilege
•Least common mechanism
•Psychological acceptability
•Minimize default attack surface
•Input validation with whitelists

### Security design review

• Individual projects ensures their code passes security tests suits.

• Akraino Stack people models individual projects, and conduct model checking (using dafny ) for fault tolerance and information flow properties.

### Security architecture

•Attack surface measurement
•Product structure or layering

### Assets & threat actors identified and addressed

### Identity and Access Management

Akraino **MUST**, if not integrated with the Operator's Identity and Access Management system, support the creation of multiple IDs so that individual accountability can be supported.
Akraino **MUST** allow the Operator to restrict access based on the assigned permissions associated with an ID in order to support Least Privilege (no more privilege than required to perform job functions).
Each architectural layer of Akraino (eg. operating system, network, application) **MUST** support access restriction independently of all other layers so that Segregation of Duties can be implemented.
Akraino **MUST NOT** allow the assumption of the permissions of another account to mask individual accountability. For example, use SUDO when a user requires elevated permissions such as root or admin.
Akraino **MUST** set the default settings for user access to deny authorization, except for a super user type of account. When Akraino is installed, nothing should be able to use it until the super user configures Akraino to allow other users (human and application) have access.
Akraino **MUST** support strong authentication, also known as multifactor authentication, on all protected interfaces exposed by Akraino for use by human users. Strong authentication uses at least two of the three different types of authentication factors in order to prove the claimed identity of a user. # Look at making this MUST for infrastructure and SHOULD for other areas #
Akraino **MUST** disable unnecessary or vulnerable cgi-bin programs. # Completely disallow cgi-bin #
Akraino **MUST** provide access controls that allow the Operator to restrict access to Akraino functions and data to authori zed entities. # Least privilege #
Akraino **SHOULD** support OAuth 2.0 authorization using an external Authorization Server.
Akraino **MUST**, if not integrated with the Operator's Identity and Access Management system, support configurable length and password expiration.
Akraino **MUST**, if not integrated with the Operator's Identity and Access Management system, support Role-Based Access Control to enforce least privilege.
Akraino **MUST**, if not integrated with the Operator's Identity and Access Management system, comply with "password complexity" policy. When passwords are used, they shall be complex and shall at least meet the following password construction requirements: (1) be a minimum configurable number of characters in length, (2) include 3 of the 4 following types of characters: upper-case alphabetic, lower-case alphabetic, numeric, and special, (3) not be the same as the UserID with which they are associated or other common strings as specified by the environment, (4) not contain repeating or sequential characters or numbers, (5) not to use special characters that may have command functions, and (6) new passwords must not contain sequences of three or more characters from the previous password.
Akraino MUST not store authentication credentials to itself in clear text or any reversible form and must use salting.
Akraino **MUST**, if not integrated with the Operator's Identity and Access Management system, support the ability to disable the userID after a configurable number of consecutive unsuccessful authentication attempts using the same userID.
Akraino **MUST**, if not integrated with the Operator's identity and access management system, authenticate all access to protected GUIs, CLIs, and APIs.
Note: I (could) read this like: "if I integrate with operator's IdAM then I don't need to do anything on authentication". Maybe improve wording…
Akraino **MUST** integrate with standard identity and access management protocols such as LDAP, TACACS+, Windows Integrated Authentication (Kerberos), SAML federation, or OAuth 2.0.
Akraino **MUST** have the capability of allowing the Operator to create, manage, and automatically provision user accounts using an Operator approved identity lifecycle management tool using a standard protocol.
Akraino **MUST** support account names that contain at least A-Z, a-z, 0-9 character sets and be at least 6 characters in length.
A failed authentication attempt **MUST NOT** identify the reason for the failure to the user, only that the authentication failed.
Akraino **MUST NOT** display "Welcome" notices or messages that could be misinterpreted as extending an invitation to unauthorized users.
Akraino **MUST** provide a means for the user to explicitly logout, thus ending that session for that authenticated user.
Akraino **MUST**, if not integrated with the Operator's Identity and Access Management system, or enforce a configurable "terminate idle sessions" policy by terminating the session after a configurable period of inactivity.

•Strong log-out and session management

## API Security

This section covers API security requirements when these are used by the Akraino. Key security areas covered in API security are Access Control, Authentication, Passwords, PKI Authentication Alarming, Anomaly Detection, Lawful Intercept, Monitoring and Logging, Input Validation, Cryptography, Business continuity, Biometric Authentication, Identification, Confidentiality and Integrity, and Denial of Service.
The solution in a virtual environment needs to meet the following API security requirements:
Akraino **SHOULD** integrate with the Operator's authentication and authorization services (e.g., IDAM).Note: Should specify explicitly what has to be supported.
Akraino **MUST** implement the following input validation control: Check the size (length) of all input. Do not permit an amount of input so great that it would cause Akraino to fail. Where the input may be a file, Akraino API must enforce a size limit.
Akraino **MUST** implement the following input validation controls: Do not permit input that contains content or characters inappropriate to the input expected by the design. Inappropriate input, such as SQL expressions, may cause the system to execute undesirable and unauthorized transactions against the database or allow other inappropriate access to the internal network (injection attacks).
Akraino **MUST** implement the following input validation control on APIs: Validate that any input file has a correct and valid Multipurpose Internet Mail Extensions (MIME) type. Input files should be tested for spoofed MIME types.

## Cryptograph

This section covers Akraino cryptography requirements that are mostly applicable to encryption or protocol methods.
Akraino **SHOULD** support an automated certificate management protocol such as CMPv2, Simple Certificate Enrollment Protocol (SCEP) or Automated Certificate Management Environment (ACME).Notes:- Possibly also: Akraino should support installing certificates as part of configuration data, ie "offline enrollment"?- For Akraino we could list explicitly which protocols MUST be supported- Security architecture to define that only AAF certman need to support interface to CA (I think)
Akraino **SHOULD** provide the capability to integrate with an external encryption service.Note: Security architecture work: Which use cases, which protocol (s)
Akraino **MUST** use symmetric keys of at least 112 bits in length.
Akraino **MUST** use asymmetric keys of at least 2048 bits in length.
Akraino **MUST** provide the capability to configure encryption algorithms or devices so that they comply with the laws of the jurisdiction in which there are plans to use data encryption.
Akraino **MUST** provide the capability of allowing certificate renewal and revocation.
Akraino **MUST** provide the capability of testing the validity of a digital certificate by validating the CA signature on the certificate.
Akraino **MUST** provide the capability of testing the validity of a digital certificate by validating the date the certificate is being used is within the validity period for the certificate.
Akraino **MUST** provide the capability of testing the validity of a digital certificate by checking the Certificate Revocation List (CRL) for the certificates of that type to ensure that the certificate has not been revoked.
Akraino **MUST** provide the capability of testing the validity of a digital certificate by recognizing the identity represented by the certificate - the "distinguished name".
Akraino **MUST** support HTTP/S using TLS v1.2 or higher with strong cryptographic ciphers.
Akraino **MUST** support the use of X.509 certificates issued from any Certificate Authority (CA) that is compliant with RFC5280, e.g., a public CA such as DigiCert or Let's Encrypt, or an RFC5280 compliant Operator CA.Note: Akraino provider cannot require the use of self-signed certificates in an Operator's run time environment.

Akraino **SHOULD** use AES for symmetric enc/dec.

Akraino **SHOULD** use RSA for asymmetric enc/dec and signatures.

Akraino **MUST** use SHA-256 or better for hashing and message-authentication codes.

Akraino **MUST** support certificate revocation.

Akraino **MUST** limit lifetimes for symmetric keys and asymmetric keys without associated certificates.

Akraino **MUST** support cryptographically secure versions of SSL (**must not** support SSL v2).

Akraino **SHOULD** use cryptographic certificates reasonably and choose reasonable certificate validity periods.

Akraino **SHOULD** check the Common Name attribute to be sure it matches the host with which you intended to communicate.

## Security Analysis

This section covers Akraino security analytics requirements that are mostly applicable to security monitoring. The Akraino Security Analytics cover the collection and analysis of data following key areas of security monitoring:

- Anti-virus software
- Logging
- Data capture
- Tasking
- DPI
- API based monitoring
- Detection and notification
- Resource exhaustion detection
- Proactive and scalable monitoring
- Closed loop monitoring
- Interfaces to management and orchestration
- Malformed packet detections
- Service chaining
- Dynamic security control
- Dynamic load balancing
- Connection attempts to inactive ports (malicious port scanning)

The following requirements of security monitoring need to be met by the solution in a virtual environment.
Akraino **MUST** support Real-time detection and notification of security events.
Akraino **MUST** support Integration functionality via API/Syslog/SNMP to other functional modules in the network (e.g., PCRF, PCEF) that enable dynamic security control by blocking the malicious traffic or malicious end users.Note: PCRF, PCEF are not good examples here à to be changed or removed
Akraino **MUST** support API-based monitoring to take care of the scenarios where the control interfaces are not exposed, or are optimized and proprietary in nature.
Akraino **MUST** support detection of malformed packets due to software misconfiguration or software vulnerability, and generate an error to the syslog console facility.
Akraino **MUST** support proactive monitoring to detect and report the attacks on resources so that Akraino's and associated VMs can be isolated, such as detection techniques for resource exhaustion, namely OS resource attacks, CPU attacks, consumption of kernel memory, local storage attacks.
Akraino **SHOULD** operate with anti-virus software which produces alarms every time a virus is detected.
Akraino **MUST** protect all security audit logs (including API, OS and application-generated logs), security audit software, data, and associated documentation from modification, or unauthorized viewing, by standard OS access control mechanisms, by sending to a remote system, or by encryption.
Akraino **MUST** log successful and unsuccessful authentication attempts, e.g., authentication associated with a transaction, authentication to create a session, authentication to assume elevated privilege.
Akraino **MUST** log logoffs.
Akraino **MUST** log starting and stopping of security logging.
Akraino **MUST** log success and unsuccessful creation, removal, or change to the inherent privilege level of users.
Akraino **MUST** log connections to the network listeners of the resource.
Akraino **MUST** log the field "event type" in the security audit logs.
Akraino **MUST** log the field "date/time" in the security audit logs.
Akraino **MUST** log the field "protocol" in the security audit logs.
Akraino **MUST** log the field "service or program used for access" in the security audit logs.
Akraino **MUST** log the field "success/failure" in the security audit logs.
Akraino **MUST** log the field "Login ID" in the security audit logs.
Akraino **MUST NOT** include an authentication credential, e.g., password, in the security audit logs, even if encrypted.
Akraino **MUST** detect when its security audit log storage medium is approaching capacity (configurable) and issue an alarm.
Akraino **MUST** support the capability of online storage of security audit logs.
Akraino **MUST** activate security alarms automatically when a configurable number of consecutive unsuccessful login attempts is reached.
Akraino **MUST** activate security alarms automatically when it detects the successful modification of a critical system or application file.
Akraino **MUST** activate security alarms automatically when it detects an unsuccessful attempt to gain permissions or assume the identity of another user.
Akraino **MUST** include the field "date" in the Security alarms (where applicable and technically feasible).
Akraino **MUST** include the field "time" in the Security alarms (where applicable and technically feasible).
Akraino **MUST** include the field "service or program used for access" in the Security alarms (where applicable and technically feasible).
Akraino **MUST** include the field "success/failure" in the Security alarms (where applicable and technically feasible).
Akraino **MUST** include the field "Login ID" in the Security alarms (where applicable and technically feasible).
Akraino **MUST** restrict changing the criticality level of a system security alarm to users with administrative privileges.
Akraino **MUST** monitor API invocation patterns to detect anomalous access patterns that may represent fraudulent access or other types of attacks, or integrate with tools that implement anomaly and abuse detection.
Akraino **MUST** generate security audit logs that can be sent to Security Analytics Tools for analysis.
Akraino **MUST** log successful and unsuccessful access to Akraino resources, including data.
Akraino **MUST** support the storage of security audit logs for a configurable period of time.
Akraino **MUST** have security logging for Akraino applications/services and their OSs be active from initialization. Audit logging includes automatic routines to maintain activity records and cleanup programs to ensure the integrity of the audit/logging systems.
Akraino **MUST** be implemented so that it is not vulnerable to OWASP Top 10 web application security risks.
Akraino **MUST** protect against all denial of service attacks, both volumetric and non-volumetric, or integrate with external denial of service protection tools.
Akraino **MUST** be capable of automatically synchronizing the system clock daily with the Operator's trusted time source, to assure accurate time reporting in log files. It is recommended that Coordinated Universal Time (UTC) be used where possible, so as to eliminate ambiguity owing to daylight savings time.
Akraino **MUST** log the Source IP address in the security audit logs.
Akraino **MUST** have the capability to securely transmit the security logs and security events to a remote system before they are purged from the system.
Akraino **SHOULD** provide the capability of maintaining the integrity of its static files using a cryptographic method.
Akraino **MUST** log automated remote activities performed with elevated privileges.

## Data Protection

This section covers Akraino data protection requirements that are mostly applicable to security monitoring.
Akraino **MUST** provide the capability to restrict read and write access to data handled by Akraino.
Akraino **MUST** Provide the capability to encrypt data in transit on a physical or virtual network.
Akraino **MUST** provide the capability to encrypt data on non-volatile memory. Non-volative memory is storage that is capable of retaining data without electrical power, e.g. Complementary metal-oxide-semiconductor (CMOS) or hard drives.
Akraino **SHOULD** disable the paging of the data requiring encryption, if possible, where the encryption of non-transient data is required on a device for which the operating system performs paging to virtual memory. If not possible to disable the paging of the data requiring encryption, the virtual memory should be encrypted.
Akraino **MUST** use NIST and industry standard cryptographic algorithms and standard modes of operations when implementing cryptography.
Akraino **MUST NOT** use compromised encryption algorithms. For example, SHA, DSS, MD5, SHA-1 and Skipjack algorithms. Acceptable algorithms can be found in the NIST FIPS publications (https://csrc.nist.gov/publications/fips) and in the NIST Special Publications (https://csrc.nist.gov/publications/sp).
Akraino **MUST** use, whenever possible, standard implementations of security applications, protocols, and formats, e.g., S/MIME, TLS, SSH, IPSec, X.509 digital certificates for cryptographic implementations. These implementations must be purchased from reputable vendors or obtained from reputable open source communities and must not be developed in-house.
Akraino **MUST** provide the ability to migrate to newer versions of cryptographic algorithms and protocols with minimal impact.
Akraino **MUST** support digital certificates that comply with X.509 standards.Note: Security architecture should define all the use cases for certificates
Akraino **MUST NOT** use keys generated or derived from predictable functions or values, e.g., values considered predictable include user identity information, time of day, stored/transmitted data.
Akraino **MUST** provide the capability of using X.509 certificates issued by an external Certificate Authority.
Akraino **MUST** be capable of protecting the confidentiality and integrity of data at rest and in transit from unauthorized access and modification.Note: Either as part of req, or separately: specify *how* to protect the data; can be different approach for:- keys/secrets- DBs- configuration data- logs- …

### Confidentiality

- Passwords stored on server as iterated salted hashes using bcrypt
- Remember me token: Cryptographic nonce is stored on client & bcrypt digest stored on server
- Email addresses only revealed to owner & admins
- HTTPS

### Integrity

- HTTPS
- Data modification requires authorization
- Modifications to official application requires authentication

### Availability

- Cloud & CDN deployment
- Timeout
- Can return to operation quickly after DDOS attack stops
- Login disabled mode
- Multiple backups

# Implementation

## Common types of vulnerable implementations

OWASP top 10 Vulnerabilities

- Injection (including SQL injection)
- Auth & session
- XSS (Esp. SafeBuffer)
- Insecure object references
- Security misconfiguration
- Sensitive data exposure
- Missing access control
- CSRF
- Known vulnerabilities
- Unvalidated redirect/fwd
- XXE (2017 A4)
- Insecure Deserialization (2017 A8)
- Insufficient logging and monitoring (2017 A10)

## Hardening

- Force HTTPS, including via HSTS (Http strict transport security)
- Hardened outgoing HTTP headers, including restrictive CSP
- HTTP-only Cookies
- User secure cookie over HTTPS
- CSRF token hardening
- Incoming rate limits
- Address Space Layout Randomization (ASLR)
- Harden or disable XML entity resolution
- Load DLLs securely
- Reflection and authentication relay defense
- Safe redirect, online only
- Do not use the Javascript eval() or equivalent functions
- Integer overflow/underflow
- Input validation and handling
- Encrypted email addresses
- Gravatar restricted

## Securely reuse

- Review before use
- Get authentic version
- Use package manager

### Deprecate unsafe functions

### Use approved tools

### Static code analysis

## Verification

Recommended tools:

| | Tool Name | Description | License |
|---|---|---|---|
| Static analysis | Coverity | This tool finds defects and security vulnerabilities in custom source code written in C, C++, Java, C#, JavaScript and more<br><br>**Coverity Scan** is a free static-analysis cloud-based service for the open source community | Commercial |
| | SonarQube | **SonarQube** (formerly **Sonar**)[1] is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities | GNU LGPL |
| | Veracode | Veracode provides multiple security analysis technologies on a single platform, including static analysis, dynamic analysis, mobile application behavioral analysis and software composition analysis. Evaluated by AT&T | |
| | Fortify | Used by AT&T | |
| | Helix QAC | Helix QAC is the most accurate static code analyzer for C and C++. | |
| | CodeSonar | CodeSonar performs a unified dataflow and symbolic execution analysis that examines the computation of the entire program. | |
| | MISRA | MISRA and the associated tools. Should we conform with MISRA standard? | |
| Dynamic analysis | IBM Security AppScan | Evaluated by AT&T | Commercial |
| | Fortify WebInspect | Used by AT&T | Commercial |
| | VeraCode | Veracode provides multiple security analysis technologies on a single platform, including static analysis, dynamic analysis, mobile application behavioral analysis and software composition analysis. | Commercial |
| | angr | angr is a platform-agnostic binary analysis framework. It performs<br><br>• Disassembly and intermediate-representation lifting<br>• Program instrumentation<br>• Symbolic execution<br>• Control-flow analysis<br>• Data-dependency analysis<br>• Value-set analysis (VSA)<br>• Decompilation | |
| | Valgrind | Valgrind tool suite provides a number of debugging and profiling tools. | GPLv2 |
| | KLEE | KLEE is a symbolic virtual machine built on top of the LLVM compiler infrastructure, and available under the UIUC open source license. | |
| | LLVM/Clang Sanitizers | It is a fast memory error detector. It consists of a compiler instrumentation module and a run-time library. The tool can detect the following types of bugs:<br><br>◦ AddressSanitizer (detects addressability issues) and LeakSanitizer (detects memory leaks)<br>◦ ThreadSanitizer (detects data races and deadlocks) for C++ and Go<br>◦ MemorySanitizer (detects use of uninitialized memory) | |
| | FlowDroid (Java) | FlowDroid is a *context-, flow-, field-, object-sensitive and lifecycle-aware* static taint analysis tool, it could be leveraged to scan Java Bytecode. | |
| Pen test | Metasploit Framework | The Metasploit Project is a computer security project that provides information about security vulnerabilities and aids in penetration testing and IDS signature development. | BSD |
| | OWASP Zed Attack Proxy (ZAP) | OWASP ZAP is an open-source web application security scanner. | Apache |
| | Autosploit | AutoSploit attempts to automate the exploitation of remote hosts. | |
| | Armitage | Armitage is a graphical cyber attack management tool for the Metasploit. | |

| | | | |
|---|---|---|---|
| | cisco-global-exploiter | Cisco Global Exploiter (CGE), is an advanced, simple and fast security testing tool . | |
| | BURP suite | | |
| | Postman | Browser plugin (Randy Stricklin to add details as to how to integrate with CI/CD | |
| Fuzzing test | OSS-Fuzz | OSS-Fuzz conducts continuous fuzzing of open source softwares. | Apache |
| | AFL | American fuzzy lop is a fuzzer that employs genetic algorithms in order to efficiently increase code coverage of the test cases.<br><br>https://github.com/mirrorer/afl | Apache |
| Vulnerability analysis | JFrog XRay | Used by AT&T. For container, npm, RPM, and debian etc artifacts vulnerability scan | Commercial |
| | CoreOS Clair | Clair is an open source project for the static analysis of vulnerabilities in application containers (currently including appcand docker ). | Apache |
| | Cybellum | Cybellum V-Ray ™. Gives full component visibility and risk assessment, based on automated vulnerability detection. | |
| | GrammaTech CodeSonar | Source code and binary level static analysis | |
| | ClamAV | Anti-virus | Open source |
| | NMAP | Discover hosts and services on a computer network by sending packets and analyzing the responses. | Modified GPLv2 |
| | OpenVAS | The OpenVAS scanner is a comprehensive vulnerability assessment system that can detect security issues in all manner of servers and network devices. | |
| | Wireshark | Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. | |
| | Nessus Professional | Nessus helps the security pros on the front lines quickly and easily identify and fix vulnerabilities - including software flaws, missing patches, malware, and misconfigurations. | |
| | John the Ripper | John the Ripper is a free password cracking software tool. | |
| Stress Test | SlowHTTPTest | SlowHTTPTest is a highly configurable tool that simulates some Application Layer Denial of Service attacks by prolonging HTTP connections in different ways. | Apache |
| | MoonGen with DPDK | Fast and flexible packet generator for 10 Gbit/s Ethernet and beyond. MoonGen uses hardware features for accurate and precise latency measurements and rate control. | MIT |
| | Pktgen with DPDK | **Pktgen is a traffic generator powered by Intel's DPDK at 10Gbit wire rate traffic with 64 byte frames.** | |
| Full stack test | Lynis<br><br>OpenSCAP | Lynis is a free and open source security and auditing tool.<br><br>OpenSCAP is used by OPNFV for security scan<br><br>Kali Linux is a Linux distribution containing 300+ penetration tool. This distribution can be used for testing external attacks on Akraino systems.<br><br>TODO: For a complete system test we may need to define a test diagram similar to this https://insights.sei.cmu.edu/sei_blog/07092018_testingtools_scanlon_figure2_2.png which should encompass all our tests and describe tests hierarchy and test strategy. | GPLv3<br><br>GPLv2 |
| Platform | | Currently, there are no open source tools for platform security verification available for Arm platforms. Arm provides Server Base Security Guide (SBSG) which is specifying security requirements and guidance for SBSA/SBBR class systems: https://developer.arm.com/docs/den0086/latest | |

# Release

## Incident Response Plan

- An identified sustained engineering team

- On-call contacts with decision-making authority
- Security servicing plan for code inherited from other group
- Security servicing plan for licensed 3rd-party code

### Final security review

Examination of

- Threat models
- Exception requests
- Tool output
- Performance against the previously determined quality gates or bug bars

FSR outcome

- Passed FSR
- Passed FSR with exceptions
- FSR with escalation

### Release/Archive

- Certify
- Archive all pertinent information and data

## Response

Security servicing and response execution

## References


1. Microsoft Security Development Lifecycle (SDL) Process Guidance - Version 5.2
2. Core Infrastructure Initiative best practices badge - security
3. ONAP Security Best Practices