IDS WG Meeting Minutes
August 22, 2024

This IDS WG Meeting was started at approximately 3:00 pm ET on August 22, 2024.

Attendees

Smith Kennedy	HP
Jeremy Leber	Lexmark
Alan Sukert	

Agenda Items

1. The topics to be covered during this meeting were:
   - Review of the HCD iTC and HIT Meetings since our last HCD iTC Meeting on 8/7/24
   - Presentation by Al Sukert on NIST SP800-218A Secure Software Development Practices for Generative AI and Dual-Use Foundation Models
2. Meeting began by stating the PWG Anti-Trust Policy which can be found at https://www.pwg.org/chair/membership_docs/pwg-antitrust- policy.pdf and the PWG Intellectual Property Policy which can be found at https://www.pwg.org/chair/membership_docs/pwg-ip-policy.pdf.
3. Al then provided a summary of what was covered at the HIT Meeting on 8/12/24 and any updates from the HCD iTC Meeting on 8/5/24.
   - The Technical Recommendation (TR) has been prepared for HCD-IT #25: RFI on SBT_EXT_EXT.1 Root of Trust - immutability and valid protection mechanisms. The process for approving TRs by the HIT and forwarding them to the full HCD iTC have been solidified; they now need to be documented in the HIR Procedures document.
   - JISEC (the Japanese Scheme) provided some guidelines for the Secure Boot SFR that the HIT reviewed. The guidelines dealt with making the SFR more general and specifying crypto functions in the SFR. The HIT decided to discuss these guidelines more at its next meeting.
   - The HIT discussed two new issues:
     - #360 FCS_IPSEC_EXT.1.10 RFI - Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:
       a. Test 1: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this cPP is used, and that the length of the nonces meet the stipulations in the requirement.
       b. Test 2: If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this cPP is used, and that the length of the nonces meet the stipulations in the requirement.
       Issue:
       Tests 1 and 2 appear to be TSS requirements rather than testing activities.
       The HIT agreed it was a valid issue that should be fixed in the SD
     - #361 - Multiple immutable roots of trust - Would it be acceptable to have multiple immutable roots of trust, any one of which could be used to verify firmware integrity?

       The HIT determined that this issue needed to be referred to the HCD iTC Hardware Verification Subgroup.

- The CC:2022 Transition Subgroup also met on 8/19/22. The subgroup looked at a document from Ohya-san of ToshibaTec that analyzed the new SFRs in CC:2022 to see if they should be included in the HCD cPP. The one new SFR in CC:2022 outside of the crypto SFRs that the subgroup felt should seriously be considered for conclusion in the HCD cPP is **FDP_SDC.1 Stored data confidentiality**. It was felt that an extended component based on **FDP_SDC.1** could adequately replace **FDP_DSK_EXT.1 Extended: Protection of Data on Disk**.

  A couple of areas that this subgroup will look at in future meetings are the Extended Components in HCD cPP against CC:2022 Part 2 SFRs and the dependencies in the HCD cPP SFRs vs. The CC:2022 Part 2 SFRs.

4. Al then went through a presentation he put together on NIST SP800-218A Secure Software Development Practices for Generative AI and Dual-Use Foundation Models. This document was essentially an update of NIST SP 800-218, Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities to incorporate recommendations, considerations and notes related to the various tasks in NIST SP 800-218.

   The link to NIST SP 800-218A can be found at https://csrc.nist.gov/pubs/sp/800/218/a/ipd. The slides for this presentation are located at https://ftp.pwg.org/pub/pwg/ids/Presentation/SSDF AI.pdf.

   - The purpose of NIST SP 800-218A is to augment the practices and tasks defined in NIST SP 800-218 Version 1.1 by adding recommendations, considerations, notes, and informative references that are specific to generative AI and dual-use foundation model development.
   - The scope of NIST SP 800-218A is:
     - *AI model development*, which includes data sourcing for, designing, training, fine-tuning, and evaluating AI models, as well as incorporating and integrating AI models into other software

       Consistent with NIST SP 800-218 Version 1.1 and EO 14110, **practices for the deployment and operation of AI systems with AI models are out of scope**
     - Similarly, while cybersecurity practices for training data and other forms of data being used for AI model development are in scope, the rest of the data governance and management life cycle is out of scope.
   - The sources of expertise used to develop NIST SP 800-218A were:
     - NIST research and publications on trustworthy and responsible AI, including the *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*
     - NIST's *Secure Software Development Framework (SSDF) Version 1.1*
     - NIST general cybersecurity resources, including *The NIST Cybersecurity Framework (CSF) 2.0*
     - AI model developers, AI researchers, AI system developers, and secure software practitioners from industry and government with expertise in the unique security challenges of AI models and the practices for addressing those challenges

   Smith asked what the definitions of Generative AI and Dual-Use Foundation Models were. Al forgot to put a slide with these definitions in the presentation, but from Appendix A: Glossary on NIST SP 800-218A the applicable definitions are:

   - **artificial intelligence:** A machine-based system that can, for a given set of human-defined objectives, make predictions, recommendations, or decisions influencing real or virtual environments.
   - **artificial intelligence model:** A component of an information system that implements AI technology and uses computational, statistical, or machine-learning techniques to produce outputs from a given set of inputs

- **dual-use foundation model:** An AI model that is trained on broad data; generally uses self-supervision; contains at least tens of billions of parameters; is applicable across a wide range of contexts; and that exhibits, or could be easily modified to exhibit, high levels of performance at tasks that pose a serious risk to security, national economic security, national public health or safety, or any combination of those matters, such as by:
  - (i) substantially lowering the barrier of entry for non-experts to design, synthesize, acquire, or use chemical, biological, radiological, or nuclear (CBRN) weapons;
  - (ii) enabling powerful offensive cyber operations through automated vulnerability discovery and exploitation against a wide range of potential targets of cyber-attacks; or
  - (iii) permitting the evasion of human control or oversight through means of deception or obfuscation. Models meet this definition even if they are provided to end users with technical safeguards that attempt to prevent users from taking advantage of the relevant unsafe capabilities.

  So, a dual-use foundation model is just an AI model based on a very large quantity of data
- **generative artificial intelligence:** The class of AI models that emulate the structure and characteristics of input data in order to generate derived synthetic content. This can include images, videos, audio, text, and other digital content.

NIST SP 800-218A next discusses the following AI uses of the Community Profiles that were introduced in NIST SP 800-218:

- AI model producers, AI system producers, AI system acquirers, and others can use the SSDF to foster their communications regarding secure AI model development throughout the software development life cycle
- Following SSDF practices should help AI model producers reduce the number of vulnerabilities in their AI models, reduce the potential impacts of the exploitation of undetected or unaddressed vulnerabilities, and address the root causes of vulnerabilities to prevent recurrences
- AI system producers can use the SSDF's common vocabulary when communicating with AI model producers regarding their security practices for AI model development and when integrating AI models into the software they are developing.
- AI system acquirers can also use SSDF terms to better communicate their cybersecurity requirements and needs to AI model producers and AI system producers, such as during acquisition processes

NIST SP 800-218A divides the tasks into the same four groups as NIST SP 800-218:

- **Prepare the Organization (PO):** Organizations should ensure that their people, processes, and technology are prepared to perform secure software development at the organization level.
- **Protect the Software (PS):** Organizations should protect all components of their software from tampering and unauthorized access.
- **Produce Well-Secured Software (PW):** Organizations should produce well-secured software with minimal security vulnerabilities in its releases.
- **Respond to Vulnerabilities (RV):** Organizations should identify residual vulnerabilities in their software releases and respond appropriately to address those vulnerabilities and prevent similar ones from occurring in the future.

Each of these groups in NIST SP 800-218A is divided into practices, and each practice contains the following elements which are slightly different from NIST SP 800-218:

- **Practice:** The name of the practice and a unique identifier, followed by a brief explanation of what the practice is and why it is beneficial
- **Tasks:** One or more actions that may be needed to perform a practice
- **Priority:** Reflects the suggested relative importance of each task *within the context of the profile* and is intended to be a starting point for organizations to assign their own priorities

- **Recommendations (R), Considerations (C), and Notes (N) Specific to AI Model Development:** May contain one or more items that recommend what to do or describe additional considerations for a particular task. Organizations are expected to adapt, customize, and omit items as necessary as part of the risk-based approach described in Section 2

The practices in each are as follows:

Prepare the Organization (PO)

- **Define Security Requirements for Software Development (PO.1)**: Ensure that security requirements for software development are known at all times so that they can be taken into account throughout the SDLC and duplication of effort can be minimized because the requirements information can be collected once and shared. This includes requirements from internal sources (e.g., the organization's policies, business objectives, and risk management strategy) and external sources (e.g., applicable laws and regulations)
- **Implement Roles and Responsibilities (PO.2)**: Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC
- **Implement Supporting Toolchains (PO.3)**: Use automation to reduce human effort and improve the accuracy, reproducibility, usability, and comprehensiveness of security practices throughout the SDLC, as well as provide a way to document and demonstrate the use of these practices. Toolchains and tools may be used at different levels of the organization, such as organization-wide or project-specific, and may address a particular part of the SDLC, like a build pipeline
- **Define and Use Criteria for Software Security Checks (PO.4)**: Help ensure that the software resulting from the SDLC meets the organization's expectations by defining and using criteria for checking the software's security during development
- **Implement and Maintain Secure Environments for Software Development (PO.5)**: Ensure that all components of the environments for software development are strongly protected from internal and external threats to prevent compromises of the environments or the software being developed or maintained within them. Examples of environments for software development include development, build, test, and distribution environments

Protect the Software (PS)

- **Protect All Forms of Code from Unauthorized Access and Tampering (PS.1)**: Help prevent unauthorized changes to code, both inadvertent and intentional, which could circumvent or negate the intended security characteristics of the software. For code that is not intended to be publicly accessible, this helps prevent theft of the software and may make it more difficult or time-consuming for attackers to find vulnerabilities in the software
- **Provide a Mechanism for Verifying Software Release Integrity (PS.2)**: Help software acquirers ensure that the software they acquire is legitimate and has not been tampered with
- **Archive and Protect Each Software Release (PS.3)**: Preserve software releases in order to help identify, analyze, and eliminate vulnerabilities discovered in the software after release

Produce Well-Secured Software (PW)

- **Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1)**: Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design and architecture should mitigate those risks; and justify any cases where risk-based analysis indicates that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) is key for improving software security and also helps improve development efficiency
- **Review the Software Design to Verify Compliance with Security Requirements and Risk Information (PW.2)**: Help ensure that the
  software will meet the security requirements and satisfactorily address the identified risk information

- **Confirm the Integrity of Training, Testing, Fine-Tuning, and Aligning Data Before Use (PW.3):** Prevent data that is likely to negatively impact the cybersecurity of the AI being consumed as part of AI model training, testing, fine-tuning, and aligning. [Not part of SSDF 1.1]
- **Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality (PW.4)**: Lower the costs of software development, expedite software development, and decrease the likelihood of introducing additional security vulnerabilities into the software by reusing software modules and services that have already had their security posture checked. This is particularly important for software that implements security functionality, such as cryptographic modules and protocols
- **Create Source Code by Adhering to Secure Coding Practices (PW.5)**: Decrease the number of security vulnerabilities in the software, and reduce costs by minimizing vulnerabilities introduced during source code creation that meet or exceed organization-defined vulnerability severity criteria
- **Configure the Compilation, Interpreter, and Build Processes to Improve Executable Security (PW.6)**: Decrease the number of security vulnerabilities in the software and reduce costs by eliminating vulnerabilities before testing occurs
- **Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.7)**: Help identify vulnerabilities so that they can be corrected before the software is released to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities. Human-readable code includes source code, scripts, and any other form of code that an organization deems human readable
- **Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.8)**: Help identify vulnerabilities so that they can be corrected before the software is released in order to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities and improves traceability and repeatability. Executable code includes binaries, directly executed bytecode and source code, and any other form of code that an organization deems executable
- **Configure Software to Have Secure Settings by Default (PW.9)**: Help improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings, putting it at greater risk of compromise

Respond to Vulnerabilities (RV)

- **Identify and Confirm Vulnerabilities on an Ongoing Basis (RV.1)**: Help ensure that vulnerabilities are identified more quickly so that they can be remediated more quickly in accordance with risk, reducing the window of opportunity for attackers
- **Assess, Prioritize, and Remediate Vulnerabilities (RV.2)**: Help ensure that vulnerabilities are remediated in accordance with risk to reduce the window of opportunity for attackers
- **Analyze Vulnerabilities to Identify Their Root Causes (RV.3)**: Help reduce the frequency of vulnerabilities in the future

The presentation lists the Tasks for each of the practices along with the Recommendations (**R**), Considerations (**C**), and Notes (**N**) specific to AI model development:

- **PO.1.1**: Identify and document all security requirements for the organization's software development infrastructures and processes, and maintain the requirements over time

  R1: Include AI model development in the security requirements for software development infrastructure and processes. AI noted this should be done anyways ass part od any secure software development process.

  R2: Identify and select appropriate AI model architectures and training techniques in accordance with recommended practices for cybersecurity, privacy, and reproducibility.

- **PO.1.2**: Identify and document all security requirements for organization-developed software to meet, and maintain the requirements over time – this shows the important of not only identifying

the software requirements but identifying the requirements for the software development environment.

R1: Organizational policies should support all current requirements specific to AI model development security for organization-developed software. These requirements should include the areas of AI model development, AI model operations, and data science. Requirements may come from many sources, including laws, regulations, contracts, and standards.

C1: Consider reusing or expanding the organization's existing data classification policy and processes.

N1: Possible forms of AI model documentation include data, model, and system cards

- **PO.1.3**: Communicate requirements to all third parties who will provide commercial software components to the organization for reuse by the organization's own software

  R1: Include AI model development security in the requirements being communicated for third-party software components. AI noted it is a very important that the applicable requirements get passed down to all third-party subcontractors.

- **PO.2.1**: Create new roles and alter responsibilities for existing roles as needed to encompass all parts of the SDLC. Periodically review and maintain the defined roles and responsibilities, updating them as needed

  R1: Include AI model development security in SDLC-related roles and responsibilities throughout the SDLC. The roles and responsibilities should include, but are not limited to, AI model development, AI model operations, and data science.

  N1: Roles and responsibilities involving AI system producers, AI model producers, and other third-party providers can be documented in agreements.

- **PO.2.2**: Provide role-based training for all personnel with responsibilities that contribute to secure development. Periodically review personnel proficiency and role-based training, and update the training as needed

  R1: Role-based training should include understanding cybersecurity vulnerabilities and threats to AI models and their possible mitigations.

- **PO.2.3**: Obtain upper management or authorizing official commitment to secure development, and convey that commitment to all with development related roles and responsibilities

  R1: Leadership should commit to secure development practices involving AI models.

- **PO.3.1**: Specify which tools or tool types must or should be included in each toolchain to mitigate identified risks, as well as how the toolchain components are to be integrated with each other

  R1: Plan to develop and implement automated toolchains that secure AI model development and reduce human effort, especially at the scale often used by AI models.

  N1: Ideally, automated toolchains will perform the vast majority of the work related to securing AI model development.

  N2: See PO.4, PO.5, PS, and PW for information on tool types

- **PO.3.2**: Follow recommended security practices to deploy, operate, and maintain tools and toolchains – Tools and toolchains are an integral part of generating software builds so it is important that they be properly deployed, operated and maintained

  R1: Execute the plan to develop and implement automated toolchains that secure AI model development and reduce human effort, especially at the scale often used by AI models.

  R2: Verify the security of toolchains at a frequency commensurate with risk

-

- **PO.3.3**: Configure tools to generate artifacts of their support of secure software development practices as defined by the organization

  N1: An *artifact* is “a piece of evidence” [16]. *Evidence* is “grounds for belief or disbelief; data on which to base proof or to establish truth or falsehood” [17]. Artifacts provide records of secure software development practices. Examples of artifacts specific to AI model development include attestations of the integrity and provenance of training datasets

- **PO.4.1**: Define criteria for software security checks and track throughout the SDLC – it will be interesting to see how this one gets implemented because it is not clear how or if this has been done in the past

  R1: Implement guardrails and other controls throughout the AI development life cycle, extending beyond the traditional SDLC.

  C1: Consider requiring review and approval from a human-in-the-loop for software security checks beyond risk-based thresholds.

- **PO.4.2**: Implement processes, mechanisms, etc. to gather and safeguard the necessary information in support of the criteria

  There were no recommendations, considerations, or notes for this task

- **PO.5.1**: Separate and protect each environment involved in software development

  R1: Monitor, track, and limit resource usage and rates for AI model users during model development.

  R2: Only store sensitive data used during AI model development, including production data, within organization-approved environments and locations within those environments. AI noted this recommendation is important and is often forgotten for all sensitive data. Ensuring development environments are secure is an important aspect that is too often neglected.

  R3: Protect all training pipelines, model registries, and other components within the environments according to the principle of least privilege

  C1: Consider separating execution environments from each other to the extent feasible, such as through isolation, segmentation, containment, access via APIs, or other means.

- **PO.5.2**: Secure and harden development endpoints (i.e., endpoints for software designers, developers, testers, builders, etc.) to perform development-related tasks using a risk-based approach – It will be interesting to see how the “endpoints” are defined for this task

  There were no recommendations, considerations, or notes for this task

- **PO.5.3:** Continuously monitor software execution performance and behavior in software development environments to identify potential suspicious activity and other issues. [Not part of SSDF 1.1]

  R4: Continuously monitor training-related activity in pipelines and model modifications in the model registry.

  R5: Follow recommended practices for securely configuring each environment.

  R6: Continuously monitor each environment for plaintext secrets

- **PS.1.1**: Store all forms of code – including source code, executable code, and configuration-as-code – based on the principle of least privilege so that only authorized personnel, tools, services, etc. have access

  R1: Secure code storage should include AI models, model weights, pipelines, reward models, and any other AI model elements that need their confidentiality, integrity, and/or availability protected. These elements do not all have to be stored in the same place or through the same

type of mechanism. Note: It was interesting this recommendation mentioned confidentiality, integrity, and/or availability.

R2: Follow the principle of least privilege to minimize direct access to AI models and model elements regardless of where they are stored or executed.

R3: Store reward models separately from AI models and data

- **PS.1.2:** Protect all training, testing, finetuning, and aligning data from unauthorized access and modification. [Not part of SSDF 1.1]

  R1: Continuously monitor the confidentiality (for non-public data only) and integrity of training, testing, fine-tuning, and aligning data. Al noted that this recommendation included the confidentiality and integrity of training as well as testing.

  C1: Consider securely storing training, testing, fine-tuning, and aligning data for future use and reference if feasible.

- **PS.1.3:** Protect all model weights and configuration parameter data from unauthorized access and modification. [Not part of SSDF 1.1]

  R1: Keep model weights and configuration parameters separate from training, testing, fine-tuning, and aligning data.

  R2: Continuously monitor the confidentiality (for closed models only) and integrity of model weights and configuration parameters.

  R3: Follow the principle of least privilege to restrict access to AI model weights, configuration parameters, and services during development.

  R4: Specify and implement additional risk-proportionate cybersecurity practices around model weights, such as encryption, cryptographic hashes, digital signatures, multiparty authorization, and air-gapped environments. Al indicated he liked this recommendation.

- **PS.2.1**: Make software integrity verification information available to software acquirers

  R1: Generate and provide cryptographic hashes or digital signatures for an AI model and its components, artifacts, and documentation. Al commented that for post-Quantum computing digital signatures is that area that NIAP is focusing its efforts on fist.

  R2: Provide digital signatures for AI model changes

- **PS.3.1**: Securely archive the necessary files and supporting data (e.g., integrity verification information, provenance data) to be retained for each software release

  R1: Perform versioning and tracking for infrastructure tools (e.g., pre-processing, transforms, collection) that support dataset creation and model training.

  R2: Include documentation of the justification for AI model selection in the retained information.

  R3: Include documentation of the entire training process, such as data preprocessing and model architecture.

  N1: Al models and their components may need to be added at this time to an organization's asset inventories

- **PS.3.2**: Collect, safeguard, maintain, and share provenance data for all components of each software release (e.g., in a software bill of materials [SBOM]) – SBOMs are another area of focus in the Executive Order

  R1: Track the provenance of an AI model and its components and derivatives, including the training libraries, frameworks, and pipelines used to build the model.

R2: Track AI models that were trained on sensitive data (e.g., payment card data, protected health information, other types of personally identifiable information), and determine if access to the models should be restricted to individuals who already have access to the sensitive data used for training.

C1: Consider disclosing the provenance of the training, testing, fine-tuning, and aligning data used for an AI model

- **PW.1.1**: Use forms of risk modeling – such as threat modeling, attack modeling, or attack surface mapping – to help assess the security risk for the software – nice to see inclusion of risk modeling as a task

  R1: Incorporate relevant AI model-specific vulnerability and threat types in risk modeling. Examples of these vulnerability and threat types include poisoning of training data, malicious code or other unwanted content in inputs and outputs, denial-of-service conditions arising from adversarial prompts, supply chain attacks, unauthorized information disclosure, theft of AI model weights, and misconfiguration of data pipelines. Al commented that it was good that AI-model threat and risk modeling should be included.

  C1: Consider periodic risk modeling updates for future AI model versions and derivatives after AI model release.

  C2: During risk modeling, consider checking that the AI model is not in a critical path to make significant security decisions without a human in the loop

- **PW.1.2**: Track and maintain the software's security requirements, risks, and design decisions

  There were no recommendations, considerations, or notes for this task

- **PW.1.3**: Where appropriate, build in support for using standardized security features and services (e.g., enabling software to integrate with existing log management, identity management, access control, and vulnerability management systems) instead of creating proprietary implementations of security features and services

  There were no recommendations, considerations, or notes for this task

- **PW.2.1**: Have 1) a qualified person (or people) who were not involved with the design and/or 2) automated processes instantiated in the toolchain review the software design to confirm and enforce that it meets all of the security requirements and satisfactorily addresses the identified risk information

  There were no recommendations, considerations, or notes for this task

- **PW.3.1:** Analyze data for signs of data poisoning, bias, homogeneity, and tampering before using it for AI model training, testing, fine-tuning, or aligning purposes, and mitigate the risks as necessary. [Not part of SSDF 1.1] Al noted that data poisoning is a big problem in AI modeling, so recommendations like the one below are important.

  R1: Verify the provenance (when known) and integrity of training, testing, fine-tuning, and aligning data before use

  R2: Select and apply appropriate methods for analyzing and altering the training, testing, finetuning, and aligning data for an AI model. Examples of methods include anomaly detection, bias detection, data cleaning, data curation, data filtering, data sanitization, factchecking, and noise reduction.

  C1: Consider using a human-in-the-loop to examine data, such as with exploratory data analysis techniques

- **PW.3.2:** Track the provenance, when known, of all training, testing, fine-tuning, and aligning data used for an AI model, and document which data do not have known provenance. [Not part of SSDF 1.1]

- **N1:** Provenance verification is not possible in all cases because provenance is not always known. However, it is still beneficial for security purposes to track and verify provenance whenever possible, and to track when provenance is unknown

- **PW.3.3:** Include adversarial samples in the training and testing data to improve attack prevention. [Not part of SSDF 1.1]

  R1: Use a process and corresponding controls to test the adversarial samples and put appropriate guardrails on training and testing use.

- **PW.4.1**: Acquire and maintain well-secured software components (e.g., software libraries, modules, middleware, frameworks) from commercial, open source, and other third-party developers for use by the organization's software

  R1: Incorporate relevant AI model-specific vulnerability and threat types in risk modeling. Examples of these vulnerability and threat types include poisoning of training data, malicious code or other unwanted content in inputs and outputs, denial-of-service conditions arising from adversarial prompts, supply chain attacks, unauthorized information disclosure, theft of AI model weights, and misconfiguration of data pipelines. Al noted that boundary-conditions and the types of problems noted in this recommendation are important factors to consider in AI risk modeling.

  C1: Consider periodic risk modeling updates for future AI model versions and derivatives after AI model release.

  C2: During risk modeling, consider checking that the AI model is not in a critical path to make significant security decisions without a human in the loop

- **PW.4.2**: Create and maintain well-secured software components in-house following SDLC processes to meet common internal software development needs that cannot be better met by third-party software components

  There were no recommendations, considerations, or notes for this task

- **PW.4.4**: Verify that acquired commercial, open-source, and all other third-party software components comply with the requirements, as defined by the organization, throughout their life cycles

  R1: Verify the integrity, provenance, and security of an existing AI model or any other acquired AI components — including training, testing, fine-tuning, and aligning datasets; reward models; adaptation layers; and configuration parameters — before using them. Al noted that it is always important to verify the integrity of any model AI or not – before using it

  R2: Scan and thoroughly test acquired AI models and their components for vulnerabilities and malicious content before use

- **PW.5.1**: Follow all secure coding practices that are appropriate to the development languages and environment to meet the organization's requirements

  R1: Expand secure coding practices to include AI technology-specific considerations. Al noted it will be interesting to see how what "AI technology-specific considerations" will be included in secure coding guidelines.

  R2: Code the handling of inputs (including prompts and user data) and outputs carefully. All inputs and outputs should be logged, analyzed, and validated within the context of the AI model, and those with issues should be sanitized or dropped.

  R3: Encode inputs and outputs to prevent the execution of unauthorized code

- **PW.6.1**: Use compiler, interpreter, and build tools that offer features to improve executable security

C1: Consider using secure model serialization mechanisms that reduce or eliminate vectors for the introduction of malicious content.

- **PW.6.2**: Determine which compiler, interpreter, and build tool features should be used and how each should be configured, then implement and use the approved configurations

  C1: Consider capturing compiler, interpreter, and build tool versions and features as part of the provenance tracking.

- **PW.7.1**: Determine whether code *review* (a person looks directly at the code to find issues) and/or code *analysis* (tools are used to find issues in code, either in a fully automated way or in conjunction with a person) should be used, as defined by the organization.

  R1: Code review and analysis policies or guidelines should include code for AI models and other related components.

  C1: Consider performing scans of AI model code in addition to testing the AI models

- **PW.7.2**: Perform the code review and/or code analysis based on the organization's secure coding standards, and record and triage all discovered issues and recommended remediations in the development team's workflow or issue tracking system

  R1: Scan all AI models for malware, vulnerabilities, backdoors, and other security issues in accordance with the organization's code review and analysis policies or guidelines AI noted this is just common sense.

- **PW.8.1**: Determine whether executable code testing should be performed to find vulnerabilities not identified by previous reviews, analysis, or testing and, if so, which types of testing should be used

  R1: Include AI models in code testing policies and guidelines. Several forms of code testing can be used for AI models, including unit testing, integration testing, penetration testing, red teaming, use case testing, and adversarial testing.

  C1: Consider automating tests within a development pipeline as part of regression testing where possible

- **PW.8.2**: Scope the testing, design the tests, perform the testing, and document the results, including recording and triaging all discovered issues and recommended remediations in the development team's workflow or issue tracking system

  R1: Test all AI models for vulnerabilities in accordance with the organization's code testing policies or guidelines. Again, this is just common sense

  R2: Retest AI models when they are retrained or new data sources are added

- **PW.9.1**: Define a secure baseline by determining how to configure each setting that has an effect on security or a security-related setting so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.

  There were no recommendations, considerations, or notes for this task

- **PW.9.2**: Implement the default settings (or groups of default settings, if applicable), and document each setting for software administrators

  N1: Documenting settings can be performed earlier in the process, such as when defining a secure baseline (see PW.9.1)

- **RV.1.1**: Gather information from software acquirers, users, and public sources on potential vulnerabilities in the software and third-party components that the software uses, and investigate all credible reports

R1: Log, monitor, and analyze all inputs and outputs for AI models to detect possible security and performance issues (see PO.5.3)

R2: Make the users of AI models aware of mechanisms for reporting potential security and performance issues.

R3: Monitor vulnerability and incident databases for information on AI-related concerns, including the machine learning frameworks and libraries used to build AI models

N1: In this context, "users" refers to AI system producers and acquirers who are using an AI model.

- **RV.1.2**: Review, analyze, and/or test the software's code to identify or confirm the presence of previously undetected vulnerabilities.

  R1: Scan and test AI models frequently to identify previously undetected vulnerabilities. AI noted this is something that is not done nearly as much as it should be for any software code.

  R2: Rely mainly on automation for ongoing scanning and testing, and involve a human-in-the-loop as needed.

  R3: Conduct periodic audits of AI models.

- **RV.1.3**: Have a policy that addresses vulnerability disclosure and remediation, and implement the roles, responsibilities, and processes needed to support that policy

  R1: Include AI model vulnerabilities in organization vulnerability disclosure and remediation policies.

  R2: Make users of AI models aware of their inherent limitations and how to report any cybersecurity problems that they encounter

- **RV.2.1:** Analyze each vulnerability to gather sufficient information about risk to plan its remediation or other risk response

  N1: This may include deep analysis of generative AI and dual-use foundation model input and output to detect deviations from normal behavior

- **RV.2.2:** Plan and implement risk responses for vulnerabilities.

  R1: Risk responses for AI models should consider the time and expenses that may be associated with rebuilding them

- **RV.3.1**: Analyze identified vulnerabilities to determine their root causes

  N1: The ability to review training, testing, finetuning, and aligning data after the fact can help identify some root causes

- **RV.3.2**: Analyze the root causes over time to identify patterns, such as a particular secure coding practice not being followed consistently

  There were no recommendations, considerations, or notes for this task

- **RV.3.3**: Review the software for similar vulnerabilities to eradicate a class of vulnerabilities, and proactively fix them rather than waiting for external reports

  There were no recommendations, considerations, or notes for this task

- **RV.3.4**: Review the SDLC process, and update it if appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to the software or in new software that is created.

  There were no recommendations, considerations, or notes for this task

5. **Actions:** None

Next Steps

- The next IDS WG Meeting will be May 5, 2024 at 3:00P ET / 12:00N PT. Main topics will HCD iTC and HIT status and a special topic to be determined.