IDS WG Meeting Minutes
July 28, 2022

This IDS WG Meeting was started at approximately 3:00 pm ET on July 28, 2022.

Attendees

Graydon Dodson	Lexmark
Matt Glockner	Lexmark
Jeremy Leber	Lexmark
Alan Sukert	
Mike Trent	Xerox
Bill Wagner	
Steve Young	Canon

Agenda Items

1. The topics to be covered during this meeting were:
   - Review of the HCD iTC Meetings since our last IDS WG Meeting on 7/14/22
   - Special Topic on NISTIR 8397 Guidelines on Minimum Standards for Developer Verification of Software
   - Round Table
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 provided a summary of what was covered at the HCD iTC Meetings since the last IDS Workgroup meeting on 7/14/22:
   - Since the HCD iTC had agreed at the 7/11/22 HCD iTC meeting that there would be no more changes to ether the FDP_UDU_EXT.1 or FPT_WIPE_EXT.1 SFRs, that removed the last major barrier to completing the Final Drafts of both the HCD cPP and HCD SD. The work at both the 7/18 and 7/25 HCD iTC Meetings was doing the last-minute comment resolutions/clarifications so the editors could complete the Final Drafts of both documents and get the Final Drafts out as soon as possible.

     At the 7/18 meeting it was clear that neither the HCD cPP or the HCD SD Final Drafts would be ready for public review by the planned 7/18 date. The new plan was for both documents to be ready by 7/25. Working off-line, Al and the HCD cPP editor Brian Volkoff were able to complete work on the HCD cPP Final Draft; the HCD cPP Final Draft (Version 0.13 dated 7/25/22) was made available for public review on 7/25/22. Al prepared a document listing the major changes that are included in the HCD cPP Final Draft which he went through in detail at the meeting. The full list is attached to the end of these minutes. Note that most the changes deal directly or indirectly with Cryptographic Erase and/or the two new SFRs FPT_WIPE_EXT.1 and FDP_UDU_EXT.1.

     The Final Draft of the HCD SD was still not ready as of the 7/25 HCD iTC Meeting; the hope is that it will be ready for final review by Monday August 1st. Al pointed out we are already at least two weeks behind the new schedule, so we are now talking about mid-September at the earliest for publishing Version 1.0 of the HCD cPP and HCD SD.
   - At the 7/25 meeting the HCD iTC finally began serious discussions about formulating the HCD iTC Interpretation Team, or HIT. The HIT is a small subset of the full iTC – usually around 6-8 persons – that is responsible for the "maintenance" of the current version of the HCD cPP and HCD SD while the full iTC works on the content of the next versions of the HCD cPP and HCD SD.

The HIT will serve as the first point of contact for questions, comments, possible errors, suggested changes, etc. in the current version of both documents. The role of the HIT is to review all of these issues and determine which issues the HIT can address and which issues must be elevated to the full iTC, and for the issues the HIT an address determine what the proper response is and generate/publish that response.

In the case of the HIT, that may be required very quickly after the HCD cPP and HCD SD get published. The reason is that after the documents get published Schemes can issue Position Statements; if those Position Statements indicate approval of the HCD cPP and HCD SD then they can be used for certifications of HCDs against those Schemes. The issue becomes whether Schemes will allow a transition period between the use of the current HCD PP and the new HCD cPP. The concern is that NIAP's history has been that they don't allow any type of transition. So, it is definitely possible that as soon as NIAP approves the HCD cPP and HCD SD, it will archive the HCD PP and require all HCDs to be certified against the HCD cPP/SD, maybe even certifications that are in process

What the HCD iTC has to do to get ready is to determine who will be on the HIT and develop the processes that the HIT will use to execute – things like:

- How will it determine what issues it handles and what issues have to be handled by the full iTC
- When and how often will the HIT meet – regularly, as needed, etc.
- Membership issues like what if members don't show up to meetings, how long will members be on the HIT (permanently vs. for a fixed time)
- Voting issues – by consensus, 2/3 vs. majority vote if necessary, etc.

The same types of issues that the full iTC put into the Terms of Reference. The HCD iTC plans to "borrow" heavily from what Interpretation Teams from other iTCs have done. The HCD iTC is setting up a small subgroup to develop the HIT procedures and determine who should be on the HIT.

The bottom line is that the HCD iTC over the next two months or so will be very busy doing three things:

- Addressing comments against the Final Drafts of the HCD cPP and HCD SD and eventually publishing Version 1.0 of both documents
- Setting up the HIT so it will be ready to go when Version 1.0 of the HCD cPP and HCD SD are published
- Developing the plan for what the future releases of the HCD cPP and HCD SD will be – what the time frame for the minor and major releases will be.

4. Al then went through his special topic for the meeting, which was a review of NISTIR 8397 Guidelines on Minimum Standards for Developer Verification of Software, one of the documents he mentioned at the talk he gave at the 7/14/22 IDS WG Meeting on the progress that had been made on implementing the Executive Order (EO) 14028 on Improving the Nation's Cybersecurity since it had been issued in May 2021. The slides Al presented at the meeting can be found at https://ftp.pwg.org/pub/pwg/ids/Presentation/Guidelines for Verification of SW.pdf.

   Some of the points Al made while reviewing the Guidelines were:

   - The guidelines are meant to be minimum standards**,** not "best practices" of software verification by software producers. These Guidelines are based on assumption that there is no single software security verification standard that can encompass all types of software and be both specific and prescriptive while supporting efficient and effective verification, so they are designed for each software producers to use in creating their own processes which can be very specific and tailored to the software products, technology (e.g., language and platform), toolchain, and development lifecycle model.

- The Guidelines had some different definitions for key terminology that differed from the standard definitions for these terms as follows:
  - Software: Executable computer programs
  - Testing: Any technique or procedure performed on the software itself to gain assurance that the software will perform as desired, has the necessary properties, and has no important vulnerabilities
  - Verification: Includes methods such as static analysis and code review, in addition to dynamic analysis or running programs
    - Verification assumes standard language semantics, correct and robust compilation or interpretation engines, and a reliable and accurate execution environment, such as containers, virtual machines, operating systems, and hardware. Verification may or may not be performed in the intended operational environment.
    - Includes vendor and developer testing

  The Guidelines focused on Verification vs. just testing.
- The Guidelines' "Minimum Standards for Develop Testing" encompasses the following 11 steps, which are not in the order they would be in a typical Software Development Life-Cycle or a Secure Software Development Life-Cycle. The eleven steps are as follows:

  a) **Do Threat Modeling**

  Al was glad that this was included because this is a critical first step. Threat modeling should be used early in order to identify design-level security issues and to focus verification and software needs should drive the threat modeling method(s) used. Threat modeling should be done *multiple* times during development, especially when developing new capabilities, to capture new threats and improve modeling. A key advantage of Threat Modeling is that it may reveal that certain small pieces of code, typically less than 100 lines, pose significant risk and require additional code review.

  b) **Do Automated Testing**

  Al mentioned that he joined a subgroup of the Common Criteria User's Forum (CCUF) at a CCUF Workshop about ten years ago that started looking at the use of test automation for helping testing against Protection Profiles. Test automation has become critical now, especially against the HCD PP (and will be against the HCD cPP/SD) because without automation it will be almost impossible for vendors to do the testing required in the HCD SD to verify the protocols and all the cryptographic functions in the HCD cPP.

  As to what's in the Guidelines, most of it is straightforward with guidance such as "Automated verification can be integrated into the existing workflow or issue tracking system' and 'Because verification is automated, it can be repeated often, for instance, upon every commit or before an issue is retired'.

  c) **Code-Based, or Static, Analysis**

  This is your classic Static Analysis generally using some type of code analysis tool. It can be a standards checker, a security code analyzer or one of multiple static analysis tools available in the market. Al mentioned that early in his career he had to do manual code inspections on Fortran code which was very difficult an labor-intensive, so tool-based static analysis is very important.

  In terms of the guidelines, the two main points were that stats analysis is:

  - Divided into two approaches: 1) code-based or static analysis (e.g., Static Application Security Testing—SAST) and 2) execution-based or dynamic analysis (e.g., Dynamic Application Security Testing—DAST) and

- Is recommend using a static analysis tool to check code for many kinds of vulnerabilities and for compliance with the organization's coding standards

d) **Review for Hardcoded Secrets**

It is interesting that this is here. This is essentially review for things like hardcoded passwords. Al recalled one product while he was at Xerox which was review by a French security firm which hammered Xerox over the number of hardcoded passwords. So, it is an important thing for vendors to review to make sure there are no hardcoded passwords or other hardcoded secrets that can be used by an attacker.

e) **Run with Language-Provided Checks and Protection**

Essentially this guideline is saying that if you have a compiler or interpreter or a software language that has options that enforce security or have built-in checks and protections both during development and in the software shipped, make sure you that you use them.

f) **Black Box Test Cases**

This is your classic "Black Box" testing where you treat the system has a "black box" and look for ways to attack it as a system. This includes things like Boundary Value Testing and testing various input sources to the system like web interfaces. Basically, Black Box tests are based on functional specifications or requirements, negative tests (invalid inputs and testing what the software should *not* do), denial of service and overload, input boundary analysis, and input combinations.

Key guidelines are:

- Tests cases should be more comprehensive in areas indicated as security sensitive or critical by general security principles
- If you can formally prove that classes of errors cannot occur, some of the testing described above may not be needed

g) **Code-Based Test Cases**

This is the traditional Unit Testing done by the code developer where branches, function calls, etc. are tested. Code coverage is a very important metric here as are other metrics like cyclomatic complexity.

It is interesting that the code coverage guideline in the document is that executing the test suite should achieve a minimum of 80% statement coverage. That is unusual, since most developments set a goal of 100% code coverage.

h) **Historical Test Cases**

This is really about Regression Testing - test cases created specifically to show the presence (and later, the absence) of a bug based on prior history. However, the Guidelines do state here that a better option is adoption of an assurance approach, such as choice of language, that precludes the bug entirely

i) **Fuzzing**

Fuzzing is a type of testing that has been around for a while but has never really gained mainstream acceptance. F**uzzing** or **fuzz testing** is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program. https://en.wikipedia.org/wiki/Fuzzing is a link to a good explanation of what fuzzing is.

The Guidelines state that the advantage of generality is that such tools can try an immense number of inputs with minimal human supervision. The tools can be programmed with inputs that often reveal bugs, such as very long or empty inputs and special characters.

j) **Web Application Scanning**

Since most HCDs have a web interface, Web Application Scanning become an important type of testing. Essentially the Guidelines state that developers should use some type of security testing tool or web app scanner to probe the web app for failures.

k) **Check Included Software Components**

The last step is basically to make sure you check all your Open Source and 3rd Party components for known vulnerabilities against known vulnerability databases. Also, make sure you have tools that can identify what your Open Source and 3rd Party components are.

- Lastly, the Guidelines provided some additional guidance in three other areas beyond testing:

a) **Good Software Development Practices**

The Guidelines basically said that developed should follow the recommendations in NIST Special Publication 800-218 Secure Software Development Framework (SSDF) Version 1.1**:** *Recommendations for Mitigating the Risk of Software Vulnerabilities*. The Guidelines also indicated that Enterprises with good secure development processes had the following characteristics:

- Create a culture where security is everyone’s responsibility.
- Uses tools to automate security checking, often referred to as Security as Code
- Tracks threats and vulnerabilities, in addition to typical system metrics
- Shares software development task information, security threat, and vulnerability knowledge between the security team, developers, and operations personnel

b) **Good Software Installation and Operation Practices**

This covers areas like:

- Configuration Files – Making sure software releases include secure default settings and caveats regarding deviations from those settings; security verification should include all valid settings and (possibly) assurance that invalid settings will be caught by run-time checks and the acquirer should be warned or notified that settings other than those explicitly permitted will invalidate developer’s security assertions
- File Permissions: Make sure that the principle of least privilege is established and that roles are established so that only those with the proper permissions can read, write, execute, and delete files
- Network Configuration: Make sure verification activities include checks for invalid secure network settings
- Operational Configuration: Make sure verifications are done in as close to the anticipated Operational Environment as possible and any assumptions associated with the Operational Environment that affect Verification are known

c) **Additional Software Assurance Technology**

This section just listed some advances like cloud technology that may affect software verification in the future

5. There was no Round Table for today’s meeting
6. **Actions:** None

Next Steps

- The next IDS WG Meeting will be August 11, 2022 at 3:00P ET / 12:00N PT. Main topics will be review of the HCD iTC Meetings since this meeting and preparation for the August 18th IDS Face to Face Session.
- The next IDS Face to Face Session as part of the August PWG Face to Face Meetings is set for August 18th from 10-12 ET.

MAJOR CHANGES INCLUDED IN FINAL PUBLIC DRAFT HCD cPP

1. **To include Cryptographic Erase into the HCD cPP and address concerns about** the fact that the FDP_RIP.1/* SFRs suggested to users that residual data is permanently removed from wear-leveling storage devices (e.g., SSDs), when in fact FDP_RIP.* can't be used for operations involving Cryptographic Erase (CE) because the actual data is still present in encrypted form, and future technologies might be capable of breaking the encryption the following was done:

   - **Replaced SFR FDP_RIP.1/Overwrite** Subset residual information protection with a new SFR FDP_UDU_EXT.1 User.DOC Unavailable **that (1) provides the option for Overwrite for the SFR to apply to both wear-levelling and non-wear-levelling storage devices and (2) to include destruction of cryptographic keys as well as overwrite to make USER.DOC unavailable.**
   - **Replaced SFR** FDP_RIP.1/Purge Subset residual information protection with a new SFR FPT_WIPE_EXT.1 Data Wiping that requires that customer-supplied D.USER and D.TSF data stored in non-volatile storage be made unavailable using Cryptographic Erase as a mandatory method and optionally using none or one or more of five other methods – overwrite, block erase, media specific eMMC method, media specific ATA erase method, or media specific NVMe method.
   - **Added or modified wording addressing Cryptographic Erase or destruction of cryptographic keys in the following Sections:**
     a. **Section 1.4.2** USE CASE 2: Conditionally Mandatory Use Cases, Item 4. Nonvolatile Storage Devices
     b. Section 1.4.3 USE CASE 3: Optional Use Cases, Item 2. Redeploying or Decommissioning the HCD
   - Added the following statement to the definition of O.STORAGE_ENCRYPTION in Section 3.5.4 Storage Encryption: "…and the TOE shall provide a function that an authorized administrator may destroy encryption keys or keying material if the TOE supports a function for removing the TOE from its Operational Environment".
   - Added the following note to Section 3.5.7 Wipe Data (optional): Note: Cryptographic erase which is covered in the mandatory requirement of FCS_CKM_EXT.4 and FCS_CKM.4 can be used as a method to remove some parts of User Data and TSF Data, but it cannot be a single method to remove User Data and TSF Data unless all the data are encrypted.
   - Because of the new FDP_UDU_EXT.1 SFR, modified Section 3.5.6 Image Overwrite (optional) to remove the statement "or by destroying its cryptographic key" in the last sentence since it was no longer necessary.
   - Changed the title of Section 3.5.7 from Purge Data (optional) to Wipe Data (optional) reflect the new FPT_WIPE_EXT.12 Data Wiping SFR

- Changed the title of Section 4.1.13 from Purge Data (optional) to Wipe Data (optional) reflect the new FPT_WIPE_EXT.12 Data Wiping SFR
- Changed the Organizational Security Policy (OSP) O.PURGE_DATE to O.WIPE_DATA to reflect the new FPT_WIPE_EXT.12 Data Wiping SFR
- Modified the Application Note for the SFR FDP_DSK_EXT.1 Protection of Data on Disk to state that if additional data other than D.USER.DOC and D.TSF.CONF are encrypted, it will be purged by the cryptographic erase process

2. Modified SFRs FPT_SBT_EXT.1.5 and FPT_SBT_EXT.1.6 for Secure Boot to clarify that they apply only to Hardware Roots of Trust.

3. Removed the previous Software Functional Requirements table that was in Appendix H: SFR List, as well as the entire appendix, that mapped SFRs to OSPs. Replaced this table with a new table in Section 5.12 TOE Security Functional Requirements Rationale that maps OSPs to SFRs and provides the rationale for that mapping.

4. Moved SFR FCS_CKM.1/AKG Cryptographic Key Generation (for asymmetric keys) from a Conditionally Mandatory to an Optional requirement.

5. Added missing or incorrect SFR Mapping Information for several SFRs.

6. Removed the Consistency Rationale Appendix as being repetitive and no longer needed.

7. The term File Encryption Key (FEK) was incorrectly used in several places in the document; it was replaced by “BEV or DEK”. Also, is some instances “DEK” was missing when it should have been included, so in those instances “BEV” was changed to “BEV or DEK” also.

8. Corrected a typo in Section 5.4.2. FDP_ACF.1 Security attribute based access control, Table 5. D.USER.JOB Access Control SFP, where “log’ should have been “job”.

9. Addressed the following NIAP Technical Decisions:
    - TD0642: FCS_CKM.1(a) Requirement; P-384 keysize moved to selection
    - TD0636: NIT Technical Decision for Clarification of Public Key User Authentication for SSH
    - TD0631: NIT Technical Decision for Clarification of public key authentication for SSH Server

10. Fixed several grammatical and typographical errors in the document.