Hardware Security Risk Assessment: A Case Study

Brent Sherman
Intel Corporation
Security Center of Excellence Group
Hillsboro, OR

David Wheeler
Intel Corporation
Security Center of Excellence Group
Hillsboro, OR

***Abstract*—The security demands on development teams are growing in direct proportion to the security incidents discovered and leveraged in computer crime and cyber warfare every day. There is ongoing research to increase the effectiveness of security defect detection and penetration testing of products, but where the literature is thin, is in actual case studies that apply security assurance processes in a large-scale hardware-centric environment. This paper adds to the literature by providing an actual case study of hardware security assurance practices using a sample size of 151 projects. Furthermore, it documents and analyzes the efficacy of deploying selective automation using quantitative weighted risk ratings of the Security Development Lifecycle (SDL) to hardware projects, including strategic reuse of existing SDL collaterals for derivative projects. The evaluated methodology provided acceptable accuracy and labor savings, but the results indicate that automation focusing on assignment of a quantitative risk scoring introduces a dilution of real security concerns; instead, an approach using qualitative analysis and assignment of security assurance tasks is more beneficial.**

I. Introduction

Ransomware, Zero Day Exploits, and massive enterprise breaches are a few of the dark headlines that have propelled security assurance to the forefront of all things digital. Juxtaposed to this need for product security is global competitiveness, quality targets, reuse demands, and shrinking time-to-market (TTM) schedules. Security assurance is a necessity that is, more often than not, resource and schedule limited.

At Intel, product teams use the SDL [5] process to ensure the design of products are built with adequate security assurance. In an ongoing drive to improve security assurance processes and reduce resource demands on projects, while also increasing the efficacy of security assurance efforts, Intel's Security Center of Excellence (SeCoE) performed a methodology trial for a Security Risk Assessment tool and security assurance process. This methodology has been tailored to an environment where there is significant reuse of dozens of silicon ingredients for new product versions and derivative products.

Hardware product teams face the problem of how to best capitalize on prior SDL work for reuse in derivatives, while process quality experts want to ensure the SDL process was properly followed for each product or platform. Historically, product teams have trouble with reuse because even when products are derivatives of each other, TTM and resource pressures force teams to trim activities that do not directly contribute to the product at hand. While it is easy to hand out the platitudes, including "Don't be short-sighted", there are real product and commercial pressures that must be addressed. This paper not only shares the results of Intel's case study on the effectiveness of security assurance automation in SDL, but also discusses the myriad of other dynamics that are driving these changes. The proposed methodology turns Intel's product-centered methodology upside-down, and focuses security assurance activities on standalone hardware components, commonly referred to as "Intellectual Property" (IP).

II. Background and Related Work

The first phase of SDL is the Requirements Phase or S0, and it includes a security risk assessment [5]. Computer security risk assessment, threat analysis, and security requirement definition and analysis have been the subject of academic research and practical engineering pursuits since the early 1970's [1]. Landwehr [3] provides an early survey on formal security models used in government and military work, and in [3] describes use of the Orange book to analyze systems and set information system security requirements. Although this literature is targeted to information systems, it provided the groundwork for defining the methodologies used in secure hardware design and SDL. Landwehr [3] formalized the early practices of security risk assessment as three steps: extract risk factors; quantify risk levels; and map to security requirements and countermeasures. Baskerville [1] discussed risk assessment methodology in three generations: basic checklists, mechanistic analysis, and logical transformational analysis. The checklist approaches were not methodical, and merely mapped known solutions (countermeasures) onto potential risks in the system. Mechanistic analysis evolved from the need to analyze more complex systems, and used a top-down, decomposition of system designs to customize security at the functional level. Logical transformational analysis uses abstract modeling to apply risk analysis across different problem domains. As the complexities of the systems grew, so did the desire to automate the analysis [3] [1].

Huang [2] outlined a two-level questionnaire methodology to prune hardware features from security analysis. In their work, they introduce the idea of tripwires (questionnaire/checklist) to prevent wasted analysis on security features that have no security impact and provide a case study to demonstrate its effectiveness. This paper builds on the ideas presented in [2], modifying the approach for large scale usage.

978-1-4673-8826-9/16/$31.00

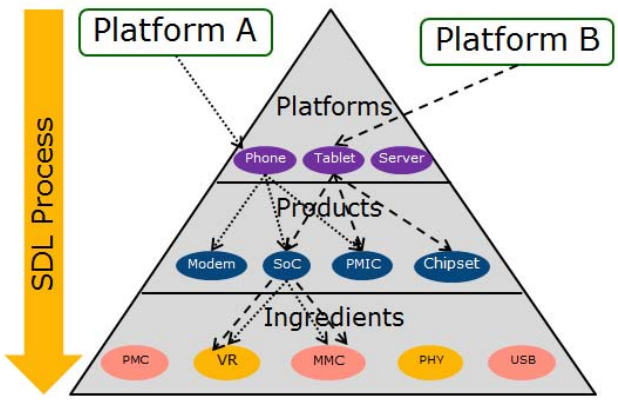

Fig. 1. Top-Down Hardware SDL Approach

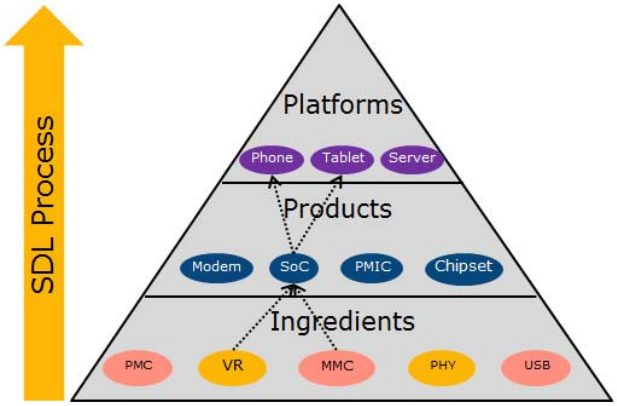

Fig. 2. Bottom-Up Hardware SDL Approach

A. Impetus for the Case Study

Intel has adopted, with some modifications, Microsoft's SDL process [5]. Relevant for this paper is the S0 phase of the process. In S0, the project planning occurs. This includes deciding the specific security assurance activities that should take place throughout the rest of the project. The participants in this phase typically include project managers, architects, designers, and a security expert. This phase can typically take between six to twenty person-hours depending on product complexity. Because hardware security experts take several years to train, they are far fewer than the number of projects and must be shared. This creates a bottleneck. The impetus for this case study came from the desire to remove this bottleneck while maintaining the quality of SDL assessments.

B. Case Study Environment

Intel SDL is typically done at a product or platform level, in a top-down fashion, where a product/platform team maintains a single SDL project, and applies security assurance activities to ingredients as needed – see Fig. 1. This is akin to the mechanistic decomposition described by Baskerville [1]. All ingredients of a platform are evaluated holistically. This is because a product team will scope their SDL activities for each ingredient based on the its usage and functionality as it relates specifically to the product's specific usage model and targeted market segment. However, this limits the reuse of SDL collaterals by other products/platforms despite the fact that many ingredients get reused across a large number of products. Fig. 1 demonstrates the reuse of ingredients across platforms A and B.

In an effort to minimize such redundancy and maximize reuse, our case study used a bottom-up approach to the SDL process, as depicted in Fig. 2. In this bottom-up approach, every ingredient goes through SDL as a standalone component. This minimizes the number of times an ingredient is evaluated under SDL[1]. Then as "products" and "platforms" integrate these ingredients, only the integration-specific points need reviewing for security concerns.

The benefit of this approach can be easily demonstrated by comparing two system on chip (SoC) examples: 1) a baseline comprised of 26 components and 2) its next generation derivative containing the same components but reused 17 without any modification. If the baseline performed a bottom-up hardware SDL on each ingredient, then the derivative team would benefit a 65% savings in their SDL process, which is quite significant. Unfortunately, this approach does exacerbate the security expert bottleneck problem mentioned earlier. While there may be a few dozen product teams in an organization, the number of IP teams are in the order of hundreds, each requiring a security expert to assist in SDL. In order to address this scaling problem, the proposed methodology details an automated approach to the S0 Phase.

III. PROCESS DESCRIPTION

The S0 phase begins with an Architect using the Security Risk Assessment (SRA) tool to determine if further SDL activities need to be performed. The tool is designed to identify security concerns by asking a series of questions. Depending on how many concerns were identified, the tool provides guidance on which SDL activities the IP team needs to perform. The concern with executing this process is the accuracy of the tool, which was validated by measuring the number of false positives and false negatives in each assessment. Therefore in parallel, a security expert was used during S0 to manually assess as the baseline against which the new tool and process were measured.

IV. SECURITY RISK ASSESSMENT TOOL

The tool was split into two sections: 1) a High-Level Filter composed of a set of questions designed to quickly filter out IPs deemed to have acceptable security risk, similar to tripwires in [2]; and 2) a Low-Level detailed assessment that contains categories based on IP subject-specific questions designed to reveal security concerns based on past security issues. However, due to the sensitivity of this topic, the specific

[1]This does not include SDL requirements for product integration.

details of the questions in the low-level assessment set are not listed; instead topic categories are briefly described.

A. High-Level Filter

At the time of deploying the tool, several IPs have already been through an S0 assessment due to early SDL adoption. The intent of this filter is to quickly identify these IPs and if there are no significant security concerns, dismiss them as minimal risk. Therefore the questions are minimal, binary (i.e. yes/no), and are not weighted. If the user answers positively[2] to any one of the questions, then that IP has *tripped* the filter and the tool will automatically guide the user to answer the Low-Level Assessment. The High-Level Filter questions are shown in Table I.

TABLE I
HIGH-LEVEL FILTER

Question	Trigger
Is the IP labeled "As Is" i.e. no modification?	No
Has the IP previously gone through the SDL process?	No
Has the IP had any security incidences in the past 2 years?	Yes

The "Trigger" column represents the answers that will *trip* the High-Level Filter. If none of the triggers are asserted, then immediately the IP is deemed to have no significant security risk and is categorized with a security risk level of *None*[3]. In this case, the IP does not have to perform any further SDL, documents the results, and continues on with the Product Lifecycle (PLC). The High-Level Filter questions were chosen specifically based on Intel's product lifecycle and current objectives, and tightly related to Intel's needs in security assurance. Other companies may have different requirements that would guide the selection of their own questions.

B. Low-Level Assessment

The Low-Level Assessment was comprised of a set of questions, organized into focused topic categories, where each question is assigned a risk rating and weighted score. The question set was not designed to be a comprehensive covering of all topic areas, but instead focuses on areas that historically had, or are perceived to have, security concerns. The questions were crafted to prevent a lengthy analysis by the user, minimize the time necessary to complete them and designed to be answered with a simple binary (i.e. yes/no) response. The complete Low-Level Assessment question set had a total of 37 questions, and all questions had to be completed prior to receiving an overall Risk Assessment score from the tool. A list of the topic areas into which the questions were separated and presented to the user is listed below; a description of each topic's focus area and intended security assurance coverage is also provided.

1) Interface connections
   - Focus: access protections, non-standard signals, security dependencies, permanent denial of service (PDoS) i.e. permanent damage to the silicon, etc.
2) Debug features
   - Focus: authorization, material exposure, bypassing any security protections, etc.
3) Firmware loading and execution
   - Focus: authentication mechanisms, patching, anti-rollback protections, etc.
4) Cryptography usage
   - Focus: compliance to NIST Cryptographic Standards and Guidelines [7], use cases, etc.
5) Memory access and protections
   - Focus: protected ranges, aliasing, decoding, Direct Memory Access (DMA), etc.
6) Power, reset, and state flows
   - Focus: shadowed registers, configuration persistence, PDoS, power protections, boot sequence, etc.
7) Privilege level support
   - Focus: privilege escalation, virtualization, etc.
8) Third party support
   - Focus: security assurance evidence

1) Calculating an Overall Risk Assessment Score: An overall Risk Assessment Score, *(R_s)*, is calculated from the answers in the Low-Level Assessment. Each Low-Level Assessment question is assigned[4] a policy-driven Risk Level, *R*, when the question was developed and added to the tool. The *R* is a value from 1 (low risk) to 5 (high risk), to create a linear risk scale. The 1-to-5 scale follows recommendations from [8] for a balanced scale, with pinning (labels) only at the anchor points, and satisfies the minimal requirement to measure individual behaviors. A larger scale was not used in order to reduce "variability without a concomitant increase in precision." [8]. However, the 1-to-5 scale is linear, and security issues generally follow a non-linear relationship. To illustrate, a vulnerability in a contained and isolated IP would not have the same impact as a vulnerability that is within an IP bridge/fabric that interacts with multiple components. In this case, the exposure makes this a non-linear relationship. To offset the linear representation, a non-linear scaling factor, *S*, is associated with each risk rating in order to accomplish this, as shown in (1).

$$W_q = S(R) \tag{1}$$

The scaling factor, *S*, is a function look-up, based on *R*, and returns the question's weighted value, W_q. Separating the Risk Level from the Scaling Factor allows the weighted

[2]Answering positively depends on the wording of the questions. Some questions are specifically worded so that a trip answer is a yes; others are worded so that a trip is a no answer. This prevents someone from merely answering no to every question.

[3]Security risk levels are explained in a later section

[4]Assigning the risk level for each question is a subjective process and out of scope of this paper.

values to be tuned during introduction and roll-out of the risk assessment tool, in order to reduce errors. The function look-up table is shown in Table II. To illustrate, assume a question was assigned a risk level of R=3, which can be considered moderate risk. Therefore if triggered, the weighted result would be W_q=S(3)=6.

TABLE II
SCALING FUNCTION LOOK-UP

Risk Defined	R	S(R)
Low	1	1
	2	3
	3	6
	4	10
High	5	16

Once all questions have been answered, the weighted results are summed together to produce the Actual Cumulative Risk, A_r, and the Maximum Possible Risk, M_r, as shown in (2) and (3) respectively.

$$A_r = \sum_{q=1}^{n} (T_q \times W_q) \tag{2}$$

$$M_r = \sum_{q=1}^{n} (W_q) \tag{3}$$

Where n is the number of questions in the assessment, T_q is the triggered response for question q, and W_q is the weighted risk score for question q. T_q is either a 1 if the user's answer triggered a security concern, or a 0 if not triggered. If not triggered, no weight is added to the cumulative risk. A_r represents the actual cumulative risks reported by triggered responses, and M_r represents the worst case possible risk if all the questions happen to trigger.

Finally, the overall Risk Assessment Score (R_s) is calculated as a percentage of the A_r from the M_r, as shown in (4).

$$R_s = \frac{A_r}{M_r} \times 100 \tag{4}$$

To demonstrate how it all fits together, suppose the Low-Level assessment contained three questions carrying the following scaled weights *S(2, 3, 4)* = [3, 6, 10]. Therefore the maximum possible risk (M_r) is 3+6+10 = 19. If the IP being evaluated triggered on questions 1 and 3 only, its actual cumulative risk (A_r) would be 3+10=13. Therefore the overall Risk Assessment Score would be R_s = (13/19) x 100 = 68%.

Once R_s is calculated, it is then used as a single metric to determine a Risk Category for treatment under SDL; a simplified 3 layer model was adopted. The risk category ranges were assigned according to Table III.

Finally, the Risk Category is used to assign the project's SDL scope, as shown in Table IV. The scope determines the security assurance activities (e.g. threat modeling, security architecture review board, detailed design analysis, static code analysis, etc.) that the IP team must execute. The higher the risk category, the more security assurance activities were assigned. This is one of the unique features of the assessment tool. Furthermore, since SDL activities directly correlate to the Risk Category, the tool was designed with an allocation range ratio of 1:3:6 [Low, Med, High] to favor more SDL with intentions to help minimize the chance of potential escapes.

TABLE III
SECURITY RISK CATEGORIES

Risk Category	Risk Assessment Level	
	Range	Allocation
Low	0-10	10%
Medium	11-40	30%
High	41-100	60%

TABLE IV
SDL SCOPE BASED ON RISK CATEGORY

Risk Category	SDL Scope
Low	Minimal: Architecture Review
Medium	Moderate: Architecture and Design Review
High	Full SDL

In actual practice, the tool provided the IP team with a list of specific Intel SDL activities. However since most company's implementation of the SDL is customized to that company's product lifecycle and development practices, the specific activities are not detailed in this paper.

C. RESULTS

Over a calendar year, IP teams used the SRA tool to evaluate 151 IPs, which are categorized in Fig 3 and had a Risk Category distribution shown in Fig 4. The 29 IPs that were evaluated as "*None*" did not trigger any of the security concerns in the High-Level Filter question set. The remaining 122 IPs were evaluated as shown, with only one IP being tagged with a *High* risk category level. Fig 5 shows the total percentage of how many times a question in a particular topic triggered a security concern. As can be seen from the chart, the top three areas of security concern were: 1) Interface connections 2) Power, reset, and state flows, and 3) Debug features[5].

Table V lists the raw results and accuracy of the tool. The accuracy was determined by comparing the results with a manual assessment done by a security expert, which was considered the baseline. The table shows the tool had an overall accuracy of 83%.

Although a Bell curve distribution was not expected, there were far fewer results in the high risk category than expected. With the scaling factor (S) exponentially increasing the risk rating for higher risk questions, and the distribution for the

[5]This does not reflect severity levels.

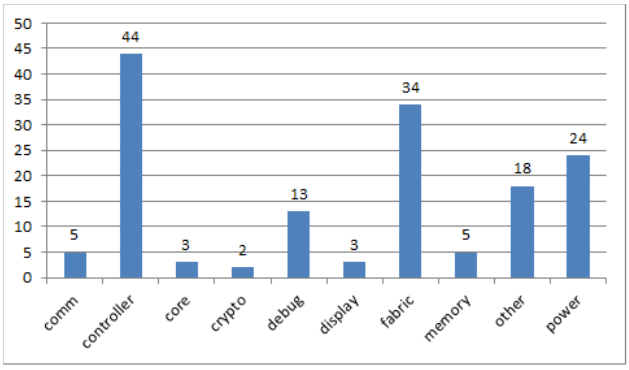

Fig. 3. IP Categorization

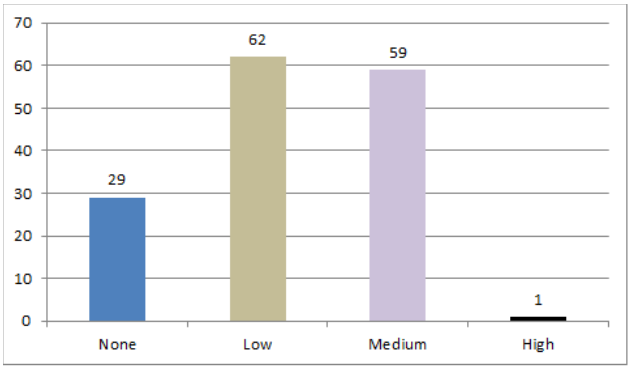

Fig. 4. Risk Category Distribution

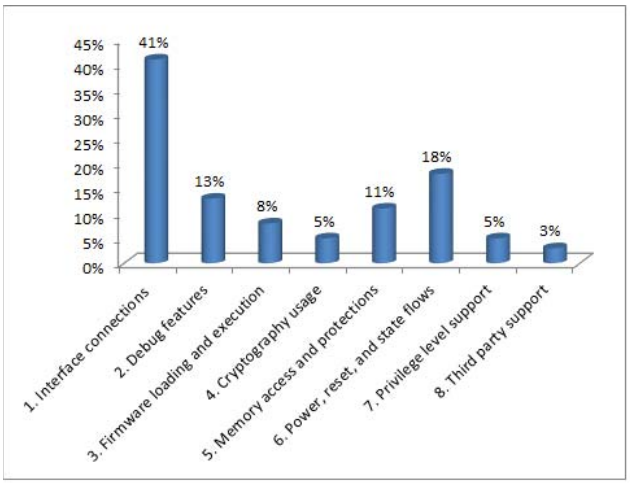

Fig. 5. Total Trigger Distribution

risk category being heavily weighted to the high category (60%), it was expected that more IP's would be required to do full SDL. This proved to be an incorrect assumption. Unless an IP had a specific security requirement (e.g. confidentiality, integrity), most were basic building blocks that only warranted minimal or moderate SDL. Nonetheless, there were still some fundamental flaws in the design of the SRA tool.

TABLE V
SRA ACCURACY

Category	Raw Results	Errors	Accuracy	Corrected Results
None	29	3	90%	26
Low	62	12	81%	61
Medium	59	10	83%	61
High	1	-	100%	3
Total	151	25	83%	151

1) Errors in Low-Level Assessment Weights: Assigning SDL activities to an overall assessment based on weights will always produce false negatives (i.e. an IP getting less security assurance activities assigned than is needed), which is very concerning. There are two reasons causing this. First, the addition of questions dilute their overall impact to the risk assessment score in equation (4). As questions get added, the value of M_r increases and using higher weights only adds to the problem. Second, each question in the Low-Level Assessment is designed to address a single known security concern. When there are multiple questions, a single trigger would never produce anything higher than a "Low" rating. If that question addressed a very serious security concern (e.g. improper use of a crypto-algorithm) than the vulnerability may never get addressed because the appropriate SDL activities were not performed.

The classification of security risk assessments (Low, Medium, High) for the purpose of assigning certain SDL activities is a failed approach, for reasons already stated. Not only does it allow for potential escapes, it also gives a false sense of security assurance. A better approach is to have the security risk assessment identify which known security concerns exist in the IP and have the SDL process directly address them.

2) Labor Impacts: The tool did make a big improvement on total time savings during the S0 phase. Being conservative, assume a manual security risk assessment takes an average of 30 minutes to complete, and at a minimum it takes a team of three individuals (pl)[6] to conduct. It would take 151 IPs a combined total of 226.5 hours to complete S0 (151 x 3pl x 0.5 hours = 226.5 hours). For our study, the SRA tool took one person on average 15 minutes to complete. This had a combined total of 38 hours (151 * 1pl * 0.25 hours). This provides a total labor savings of 188.5 hours (226.5 – 38), which yields an 83% labor reduction for S0. Table VI shows

[6]A typical team would be comprised of the SDL lead, lead architect, and a security expert.

how the bottom-up approach to SDL can impact the overall SDL effort and time savings for a product or platform.

TABLE VI
POTENTIAL SDL EFFORT SAVINGS

IP	Baseline SoC	1st Derivative	2nd Derivative
Reuse	-	124	167
Minor Change	-	22	9
New	163	33	6
Total	163	179	182
SDL Complete (ROI)			
Reuse	-	69%	92%
SDL Remaining			
Minor Change (20% mods)	-	4.4 (new)	1.8(new)
New	-	33	6
Total IP to review	-	37.4	9.8
SDL Scope	-	21%	4%

This table is an actual SoC product roadmap that shows the number of IPs going into the baseline and its two derivatives. If the baseline product teams performed SDL at the ingredient level for all 163 IPs, then any derivative reusing the IPs would benefit since the SDL process is already complete[7]. Since the 1st derivative reused 124 IPs from the baseline and the 2nd derivative reused 167 IPs from the 1st derivative, the teams only had 31% and 8%, respectively, of the IPs requiring new SDL work. Furthermore, if we assume a "Minor Change" is considered to be at worst case a 20% modification, meaning 80% of the IP has already been through the SDL process, then the overall IP SDL remaining would yield 21% for the 1st derivative and 4% for the 2nd derivative. This amounts to a significant savings in person hours for a product or platform.

3) Key Takeaways: The paper addressed two areas of concern: 1) reuse of SDL per IP and 2) SDL resource constraints at the assessment phase. Replacing the top-down methodology with the bottom-up approach yield positive results and addressed the "reuse" concern. The SRA tool addressed resource constraints by automating the assessment phase to quickly and consistently filter out IPs with acceptable risk. However, using the tool to create a single risk assessment score and assign specific SDL activities based on that score is not recommended due to the false negatives it will produce. The following is a list of the key learning points from the case study.

- Using weighted questions to determine the overall security risk assessment is not recommended. Weighting related questions within a specific topic may have value however performing an overall tally will produce false negatives. Instead, it is best to have action(s) associated with a specific question addressing a security concern (e.g. Action: The crypto-algorithm used is deemed insecure and should be replaced with one approved by NIST).
- Every question should either have a follow-on question and/or specific action(s) associated with it. This will provide better guidance on what to focus on for the rest of the SDL process. If there are no actions assigned, then it is assumed the IP had acceptable risk and is complete with the SDL process.
- Careful attention should be paid to how the questions might be interpreted, especially if the questionnaire has to support multiple demographics. Avoid any ambiguity, bias, and slang or colloquial phrases/words.
- Do not use combined or compound questions. Each question should identify a single, specific security concern, not multiple threats, even if those threats are related.
- Inclusion of use-case questions can be helpful. The IP owner might not know the market segment for their IP however they do understand how the IP should work and behave under certain conditions. For example, we found that the tool did not ask enough questions about what conditions were allowed for debug access, especially across power and state changes.

[7]This does not take into account any SDL efforts required by the integration into the SoC.

D. Conclusion

Automating a bottom-up approach to SDL has proven to be a scalable methodology, with significant labor savings, enabling faster TTM for product/platform teams. The level of security risk can be adjusted to prudent levels, as long as the automation process protects against dilution of identified threats. This case study showed some of the flaws implemented in automating a security risk assessment as a single metric to determine SDL activities and the results of such flaws. However, with the guidance of the key takeaways provided, it is believed that automating the S0 phase can be achieved with minimal or acceptable risk and become a useful tool to meet both security assurance and TTM requirements.

ACKNOWLEDGMENTS

Our thanks to Jill C. Wheeler for her contributions and recommendations.

REFERENCES

[1] Baskerville, R. 1993. Information systems security design methods: implications for information systems development. ACM Comput. Surv. 25, 4 (December 1993), 375-414. DOI= http://doi.acm.org/10.1145/162124.162127.

[2] Huang, R., and Grawrock, D., and Doughty, D., and Suh, G.E. 2011. Systematic Security Assessment at an Early Processor Design Stage. In Proceedings of the 4th international conference on Trust and trustworthy computing (TRUST'11). Springer-Verlag, Berlin, Heidelberg, pp 154-171.

[3] Landwehr, C.E. 1981. Formal Models for Computer Security, ACM Computing Surveys, Vol. 13, Number 3 (September, 1981).

[4] Landwehr, C.E, and Lubbes, H. O. 1985. An Approach to Determining Security Requirements for Naval Systems. Naval Research Lab Technical Report 8897. http://www.landwehr.org/1985-nrl-fr-8897-landwehr.pdf.

[5] Microsoft. 2010. Security Development Lifecycle: Simplified Implementation of the Microsoft SDL. Retrieved from http://www.microsoft.com/en-us/download/details.aspx?id=12379

[6] Myagmar, s., Lee, A., and Yurcik, W. Threat Modeling as a Basis for Security Requirements. In Symposium on Requirements Engineering for Information Security (SREIS) in conjunction with 13th IEEE International Requirements Engineering Conference (RE), Paris, France, August 29th, 2005.

[7] FIPS PUB 140-2, Security Requirements for Cryptographic Modules. Retrieved from http://csrc.nist.gov/groups/STM/cmvp/standards.html

[8] Friedman, H., Amoo, T. 1999. Rating the Rating Scales. Journal of Marketing Management, Vol 9:3, Winter 1999, 114-123.