RC 25035 (W1008-039), 11 Aug 2010
Computer Science

IBM Research Report

Lessons Learned Building the Caernarvon High-Assurance Smart Card Operating System

Paul A. Karger, Suzanne K. McIntosh, Elaine R. Palmer, David C. Toll, and Samuel M. Weber

IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598, USA

Research Division
Almaden – Austin – Beijing – Delhi – Haifa – India – T.J. Watson – Tokyo – Zurich

This paper has been submitted to IEEE Security and Privacy Magazine.

Lessons Learned Building the Caernarvon High-Assurance Smart Card Operating System

Paul A. Karger, Suzanne K. McIntosh, Elaine R. Palmer,
David C. Toll, and Samuel Weber[1]
IBM Corporation, Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
karger@watson.ibm.com, (skranjac,erpalmer,toll)@us.ibm.com

ABSTRACT

In this paper we share lessons learned in designing, developing, and testing features for a high-assurance smart card operating system. In particular, this paper describes our software design, development, and testing processes, and the advantages reaped from following established process guidelines. We describe the project impact experienced from external influences and count among them market pressure from a rapidly changing commercial landscape which demands agility in order to assure continued funding and product success.

1 Introduction

The Caernarvon smart card operating system is a high assurance operating system designed and built with security in mind from the start. It is targeted at Evaluation Assurance Level EAL7, the highest possible level of the Common Criteria, the international standard for evaluating the security components of information systems [3]. High assurance systems like Caernarvon are useful in commercial, government, and military applications, where valuable assets or human lives must be protected from accidents and malicious attacks.

The operating system was named "Caernarvon"[2] after the magnificent castle in North Wales (See Figure 1), which was built at the end of the thirteenth century and is still standing. Construction of the original Caernarvon lasted 47 years [23]. Construction of its electronic namesake was also expected to be an arduous and lengthy development effort.

A Caernarvon smart card is designed to withstand hardware and software attacks, and continue to function correctly even when the client system has been compromised. Security critical functions include establishment of one end of strong two-party authentication, digitally signing sensitive transactions, protecting and processing confidential data without leaking information, and performing sensitive cryptographic operations that cannot be entrusted to the host client.

The Caernarvon operating system was designed from the start to be evaluated under the Common Criteria [3] at EAL7, the current highest level of assurance. Levels EAL6 and EAL7 are often called the *high assurance* levels, because systems evaluated at those levels under the Common Criteria are the only systems that have been shown to be generally resistant against

[1] Now with the National Science Foundation, 4201 Wilson Boulevard, Arlington, Virginia 22230, sweber@nsf.gov
[2] In Welsh it is pronounced approximately "kire-NAR-fon"; in English, "KAR nar vən."

sophisticated penetration attacks. These attacks are currently commonplace, but previously were considered only the concern of the military , and include such problems as buffer overflows, incomplete argument validation, spyware, Trojan horses, and root kits. High assurance is specifically designed to address these problems.

Figure 1. Caernarvon Castle in North Wales

1.1 Motivation

Electronic identification cards, passports, and mobile phones are used in security-sensitive applications, such as assuring national security and performing banking transactions. Additional platforms, such as sensors and actuators, also share the same need for reliable and trustworthy information. These devices can provide trustworthy information if applications are run in a secure operating system such as Caernarvon. As the sensor and actuator market evolves, businesses, governments, private citizens, and autonomous systems will leverage information from local and globally distributed sensors in their decision-making process. The end-user, be it an organization, a human, or an autonomously thinking and acting piece of software running in a computer, a mobile phone, or a robot, will assume that the source of that data can be trusted to deliver reliable information. This is a dangerous assumption if the operating system that runs the sensor application is not a high-assurance operating system and becomes compromised, either through malice (by an attacker) or by accident (by a co-resident application). In addition to sensors, we are also concerned with actuators – can they be trusted to perform precisely the requested action at the indicated time if they are managed by a compromised, non-secure operating system?

1.2 Features of the Caernarvon Operating System

The Caernarvon operating system supports multiple applications from mutually distrusting (and potentially hostile) sources and provides high assurance by enforcing mandatory access controls and using hardware protection mechanisms to enforce security.

Caernarvon provides for field-downloadable applications and manages the controlled sharing of data between applications. It reduces development and evaluation costs for applications because cryptographic and other common functions such as communication and authentication are provided by Caernarvon and have been tested and verified. The cryptographic library is already Common Criteria certified at EAL5+.

A more detailed description of the major features of the Caernarvon operating system can be found in [32].

1.3 Strong Authentication

Smart card systems typically leave authentication to application programs, resulting in potential security flaws, breaches of privacy, and inconsistencies in protocols. Caernarvon takes a different approach by providing a very high quality, standardized, cryptographic authentication protocol that also protects the smart card holder's identity [31]. It relieves application developers from the task of inventing, designing, implementing, testing, and evaluating their own protocols, thereby reducing development costs. The Caernarvon privacy-preserving authentication protocol has been incorporated into the European CEN standard for digital signature applications on smart cards [1]. By using a standard protocol, it is easier for a multi-application card to work reliably and securely with applications from many different sources. Authentication is an important component of the operating system, and is within the scope of the EAL7 evaluation, thus assuring that it actually works and is secure.

1.4 Mandatory Security Policy

The Caernarvon system enforces a mandatory security policy. Each directory or file in the system's storage has an associated access class. Access is granted only to those users (programs) that have the appropriate access class, which is determined during the authentication process. Caernarvon introduces a new approach for multi-organizational mandatory access controls. It provides a mechanism for access controls between different organizations (such as Payroll vs. Purchasing or Department of Defense vs. Department of Energy). This new scheme can handle millions of different organizations with organization-specific security policies, all connected to a common Internet.

Caernarvon has a new approach for defining access classes dynamically in the field. A smart card holder can securely download new applications and access classes from application providers who were not even known to the card issuer at the time the card was issued to the card holder. This feature provides significant benefits, both in commercial applications (where, for example, an airline frequent flier program may sign on new hotel or car rental partners long after issuing a card to a customer) and in the military (where new coalitions may require access by personnel from countries with whom the issuer has never been allied before).

2 Using the Common Criteria Evaluation Process

To pass an EAL7 evaluation, the Common Criteria requires the strongest software engineering techniques known. These techniques include a formal security policy model, a full system design with a formal high-level design specification, and a formal proof of correspondence between the security policy model and the high-level design specification. It also requires a specification of all internal functions in a semi-formal low level design with a demonstration of correspondence between the high-level design, the low-level design, and the actual software code. The value of formal specifications was shown when the formal specification of the Caernarvon formal security policy model [30] actually identified a security problem in the original version of the informal security policy model.

The development cycle must include intensive design and code reviews and full configuration control. There was value derived from conducting code reviews. For example, one code review revealed a problem in the implementation of the Digital Signature Standard (DSS) as shown in Section 7.1 below. There must be comprehensive testing, including code coverage analysis that every path has been exercised, and that no dead code exists. Finally, there must be an extensive vulnerability analysis for possible security loopholes, as well as a search for covert communications channels.

Certification at a high assurance level requires that the system have comprehensive documentation - this documentation has been written and maintained from the very start of the project. The documentation for Caernarvon can be regarded as falling into a number of categories, including the system specification, system internals documentation, test system documentation, and development tool documentation. Large portions of the internals documentation and test generation are actually embedded in comments in the source code and test scripts. One of the development tools is a program that extracts the documentation from the source code and produces the necessary Common Criteria documentation in book form.

Meeting the strict documentation standards of the Common Criteria proved very helpful to us. The ISO standards for smart cards have ambiguities that all implementations have had to resolve. However, other implementations have not clearly and openly documented how those ambiguities were resolved, leading to interoperability problems between different readers and smart cards. The strict documentation requirements forced us to identify these ambiguities early and to clearly document exactly how we chose to resolve them. This will help end users and integrators understand where the problem areas lie and how they were resolved in the Caernarvon OS.

3 Collaboration Matters

> "Something magical happens when you bring together a group of people from different disciplines with a common purpose."
>
> - Mark Stefik, Fellow, Palo Alto Research Centre[3]

Creating a high assurance operating system smart card requires deep skills from multiple technical disciplines: software design and development, hardware design and development, exhaustive testing, vulnerability analysis, formal modeling, and security evaluation. Likewise, it

[3] http://www.leading-learning.co.nz/famous-quotes.html

requires skills from multiple business disciplines: market analysis and development, requirements gathering, relationship management (with customers, business partners, and executives), project management, budgeting, procuring funding, and public and media relations.

From the outset, we realized that no one organization held all the skills (nor the necessary funding) to support the entire effort on its own. Therefore, we enlisted the help of business partners, and divided the project into six nearly equal technical efforts: hardware design and implementation, software design and implementation, test framework / testing, Common Criteria documentation, formal modeling, and vulnerability / covert channel analysis. Initially, the project team spanned seven enterprises in five countries: IBM® (in New York, Kentucky, Germany, and Switzerland), Philips Semiconductors (now NXP) (in Germany and Austria), atsec Information Security, the German Federal Office for Information Security (BSI), the German Research Center for Artificial Intelligence (DFKI), the University of Augsberg, and a Common Criteria evaluation laboratory in Germany. We found it absolutely necessary that the enterprises have legal agreements in support of full disclosure of proprietary technical details. The technical rationale behind this finding of full disclosure is given in [18]. In essence, the hardware and the operating system are bound as a united front to protect the operating system from applications, and the applications from each other. Thus, there are many security-sensitive interactions among the hardware, software, formal model, Security Targets, and detailed evaluation technical reports. Details of the interactions can be gleaned from careful study of proprietary documents tightly held by each organization. Unfortunately, organizations are often bound by restrictive evaluation contracts, which actually prohibit full disclosure of this information to third parties. One of the most important lessons we learned was that information sharing among partners was critically important to the success of the project and to the security of the system we were developing.

> Good design begins with honesty, asks tough questions, comes from collaboration and from trusting your intuition.
>
> - Freeman Thomas, Director, Strategic Design, Ford Motor Company[4]

Project management was simplified by a clear delineation of duties, was carried out by each organization, and was coordinated by the global project manager. Organizations exchanged Gantt charts, which detailed their deliverables, schedules, and manpower. We held weekly conference calls, with all team members participating, during which we discussed both technical and business matters. We also held an initial face-to-face kickoff meeting (for building trust and relationships among the team members), followed by bi-annual face-to-face meetings.

The multi-organizational nature of the project helped to protect the project when budget pressures threatened the participants. Organizations were reluctant to cancel a project with contractual commitments to long-established business partners.

4 High Assurance Requires a Long Term Commitment

Building a high assurance system is like building a brick house from the old fairy tale, *The Three Little Pigs* [5] only making it earthquake, fire, and flood-resistant, as well as wolf-resistant. Such

[4] http://www.brainyquote.com/quotes/authors/f/freeman_thomas.html

an effort takes longer and has a higher upfront investment than one of lower assurance. In addition to being resilient to attacks, a rarely noted side effect is that the high assurance system is extremely reliable. (See the discussion of high-assurance robustness in section VII.b of [20].)

Unfortunately, it has been our experience that the time required to complete high assurance systems exceeds the typical funding horizons of commercial organizations. It is quite common for a high assurance project to be cancelled, just as it is nearing completion. The problem of limited horizons is not unique to commercial organizations. Government efforts to develop high-assurance projects have suffered similar problems as higher authorities abruptly cancel efforts [8, 28] that are otherwise progressing very well.

It is advisable, therefore, to divide a high assurance project into one with intermediate marketable deliverables, even though they do not attain the ultimate target level of assurance. These intermediate deliverables will provide evidence to management, customers, and, most importantly, those who fund the project, that the project is indeed on target, progressing, and within budget. To continue the analogy, it is possible to design a robust brick house, but then make carefully thought-out compromises during final construction to lower costs and expedite delivery of an interim system, for example, by designing for heavy, bullet-proof windows, but temporarily replacing them with ordinary glass, and selling the house at a lower cost. This strategy is likely to garner continued management support by generating some interim funding, so that the project can survive long enough to reach its end goal.

> Even if you are on the right track, you will get run over if you just stand there.
> - Will Rogers[6]

The resulting product is still better than others on the market, but as built, could not rate a high assurance certificate. In the case of the Caernarvon project, the business partners worked together to create an intermediate deliverable, the cryptographic library, which earned the world's first EAL5+ (the highest to have been awarded at that time) for a cryptographic library [2]. It was shipped as a product, and was a strong foundation for the remainder of the Caernarvon project. In the course of completing this interim deliverable, the team gained valuable experience with Common Criteria. Two valuable lessons learned were that the requirements at high assurance are often vague (for example, what, exactly, qualifies as a Measurable Life Cycle), and that interpretations of the requirements may change as the evaluation staff changes over the long life of the project.

Note that it is very important to stick with the original high assurance design, and simply make compromises where retrofitting is feasible later. It is not possible to build a house of straw, then attempt to strengthen it later. No matter what, it still has only straw at its foundation, and it can never be fortified into a house of bricks.

---

[5] One pig builds a house of straw, one pig builds a house of sticks, and the other pig builds a house of bricks. A wolf destroys the houses of straw and sticks, but the house of bricks remains standing, despite the onslaughts of the wolf.

[6] http://www.brainstorming.co.uk/quotes/creativequotations.html

In the long run, producing intermediate, lower-assurance deliverables takes even longer than building only one high assurance product. Later in the project, formal modeling and design reviews will inevitably expose weaknesses that must be corrected. Therefore, maintaining compatibility between the intermediate and long term deliverables may not be feasible. Furthermore, if the intermediate deliverables are no better than those of the competition, they are not worth the extra time and cost.

As evidenced by the collaboration described above, no one party need foot the entire bill, but all have to agree to stay in the game for the long haul. We have often been asked about the actual cost of our project. Although the actual costs are protected by confidentiality agreements, the relative costs of each task over a four year period are estimated as follows:

Operating System development and documentation	44%
Testing	24%
Formal Modeling	10%
Vulnerability Analysis	9%
Combined Software + Hardware evaluation	7%
Hardware documentation	6%
total	100%

5 Tools in Support of High Assurance Development

Although our primary goal was to develop a high assurance operating system, we ultimately purchased and developed a considerable amount of tooling in support of our efforts. The effort to develop new tools and extend existing ones was significant, and was measured in multiple person-years rather than in a few person-months. Although the tools required much up-front effort, they reduced the overall effort required to complete the system. Our tools enabled us to generate the complex documentation required by Common Criteria, facilitated automated testing of the system software, and helped us perform vulnerability analysis on the system software. Some of the tools are discussed briefly below, and details can be found in [33].

5.1 Design and Implementation Documentation

A product which is designed to be evaluated at a high level under the Common Criteria must necessarily have a coherent and comprehensive set of documentation. The Caernarvon project relied heavily on the robustness of FrameMaker[7], as we created and updated multiple volumes totaling several thousand pages with thousands of cross-volume cross-references (all this for a relatively small system of approximately 47,000 lines of code). From its inception, the Caernarvon project had a full specification and documentation, including (but not limited to) the high-level design, the low-level design, and the test suite.

The project coding standards required that the specification of each function be included in comments at the start of the function. The Common Criteria also required that this specification be included in the low-level documentation of the system submitted for the evaluation. Hence it was decided that much of the low-level documentation for the code (and the test suite) would be

[7] FrameMaker is a trademark of Adobe Systems Incorporated.

generated from the source code, rather than being written twice (once in the code, and once in the documentation). While this practice is common in object-oriented systems, tools to support it were not available in our development environment with specialized compilers, assemblers, and device emulators for smart cards. Simple extraction of comments was not difficult. What *was* difficult was maintaining cross references to documentation that was in other parts of the code or that wasn't in the code at all, and keeping track of the origin of the documentation.

We also solved other difficult problems associated with generating Common Criteria documentation: automatically generating lists of which functions tested other functions, which functions were tested by other functions, and getting the machine-generated information into a format suitable for importation into FrameMaker. In the end, we created (and generated) a comprehensive and professional set of documentation, with a consistent style, which included the system specifications, the low-level code documentation, and the test suite documentation, available in hardcopy and electronic format (with hyperlinks).

5.2 Tools for Development, Testing, and Analysis

One of the philosophies behind the Common Criteria is that no one technique or tool can guarantee a secure product, and as a result, multiple approaches should be used so as to reduce the chance of security flaws. Likewise, we do not claim that any particular tool or strategy (including the ones we used) is "best," and we actively recommend the use of a wide variety of security-related tools. Broadly speaking, the tools can be divided into a number of categories based on their scope, such as life-cycle approaches, test generation, and high-level design analysis.

In ordinary development projects, such tools are readily available and work satisfactorily. In our high assurance project, however, the lack of suitable commercial off-the-shelf tools in support of high assurance requirements significantly affected the process of development, prolonged the schedule, and increased the budget.

5.2.1 Development environment

We chose a smart card processor based on strong hardware features that could support a high assurance operating system, and there was no other alternative at the time we began the project. For this processor, there was only one vendor that offered a complete suite of tools for development and testing (compiler, assembler, linker, utilities, hardware emulator, and interactive development environment). When we encountered bugs in those tools, we either developed workarounds or waited until the vendor could fix the bugs. Had an open source compiler or a competitor's development suite been available, we would have had more flexibility in recovering from such unfortunate events.

5.2.2 Configuration Management System

The Common Criteria requires the use of a configuration management system (CMS) to store the implementation, the system documentation, all tools, all tests, lists of common commercial software used, and various other project-related documents. Hence we decided to use the IBM

Rational[8] ClearCase and ClearQuest products, which provided full integration of configuration management, bug reporting, and fault and change management.

5.2.3 Test Suite

Development of a system targeted at Common Criteria EAL7 requires extensive testing of the system, which in turn requires a comprehensive test suite. In our case, the test suite included thousands of tests. Furthermore, each test case verified that the state returned from the system under test matched the expected result (i.e., success or a specific error or failure). It is clearly impossible for a person to reliably check a large number of tests manually.

Our partners in Philips Semiconductors (now NXP) suggested using a test suite written in Ruby. This test suite has undergone extensive development, including facilities to exercise the internal functions of the system under test, to execute only some of the tests in a given run, and to automatically compare and verify the results of each test. Thus, to attain the goal of all tests being self-checking, it was necessary only to ensure that, as each test was written, the expected return values were specified within the test. This slightly increased the time to write the tests but avoided much of the work that would have been required if test verification code had been written separately or added later.

The Common Criteria requires compliance with cryptographic algorithm and random number generation standards. One problem with the compliance tests for these standards is that, on slow processors such as smart cards, they can take a long time to run. For example, the full NIST test suite for the DES algorithm, including all of the Monte-Carlo type tests, could take months to execute on a smart card. In such cases, it would be appropriate for the developer and the evaluation agency to agree on a suitable subset of the tests to be run.

The Common Criteria requires, at EAL7, that all code in the system be exercised by the tests. This requirement assures that there is no "dead" source code (code that can never be accessed). The vendor-provided development suite provided a facility to obtain details of all the machine instructions that were executed in a testing run, but did not provide any means of matching this information back to the program source. We developed a tool to perform this matching, for both C and assembler.

At EAL7, the Common Criteria requires that the design of the system be built in layers; function calls could be made only to the same or lower layers—upward calls were not allowed. This layering requirement was specified by Schell [29], based on the work of Parnas [26] and Dijkstra [15]. The toolset in use did not produce an adequate cross-reference of the system that could be used to verify that this requirement was met. However, we used software that produced highlighted listings of C and Assembler code and also generated a full cross-reference of the function calls in the system. A Ruby utility then processed this cross-reference file to flag any illegal upward calls.

[8]IBM, Rational, ClearCase, and ClearQuest are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

5.2.4 Formal Specification Tools

The Caernarvon formal security policy model was developed by a team from the University of Augsberg and the DFKI [30] using the VSE II specification language and proof system[17]. While not required at Common Criteria EAL7, the Caernarvon operating system assurance argument could certainly benefit from code-level proofs such as those demonstrated by the seL4 project [21].

5.2.5 Static Analysis Tools

At EAL7, the Common Criteria requires that the developer perform a systematic vulnerability analysis of the Target of Evaluation. In an attempt to automate some aspects of the vulnerability analysis, we chose to apply static analysis of the implementation to find bugs that were not discovered by the extensive testing described above. Our search for a static analysis tool led us to BEAM (short for "Bugs, Errors, And Mistakes"), developed by another group at IBM Research [10]. BEAM met our requirements: it analyzed C language source code; the BEAM source code was available to us, enabling the addition of features required by our platform; and it cost nothing for internal use within IBM. This was important, because the C compiler for the Philips smart card chip included some non-standard features that had to be handled specially.

As part of a related research project, we used BEAM to analyze the implementation of one component of our operating system, the file system, which was approximately 5,000 executable statements of C code. For this part of the project, we recruited other researchers to help us with the static analysis. They were experts with static analysis tools and understood operating system internals quite well, but were unfamiliar with the intricacies of smart cards. For this reason, we chose a relatively easy-to-understand, finite, well-defined body of code to start with. It was easy to analyze as a unit.

BEAM (with human assistance in interpreting the results) found about a dozen bugs and reported numerous violations of best practices. One common type of bug found was uninitialized variables; for example, an execution path could bypass the place where a variable was first set. Many of the data structures use short (8-bit or 16-bit) fields to conserve space, and BEAM found certain errors when performing arithmetic on these limited length fields, particularly if the operation was between a shorter and a longer variable. Our experience here indicates that, in any high-security project, it would be advisable to analyze all of the code of the system under development.

5.2.6 Test Generation

Common Criteria requires the application of a careful and systematic test methodology.
Due to the lack of test generation tools for the system, the tests for the Caernarvon system were written by hand. Note that certain tests, in particular those for cryptographic algorithms, can be obtained from sources such as NIST websites. However, these tests require a program to convert the tests into a form that can be executed by the test system.

Manually generated tests have historically been found to be poor, and therefore test generation methodology has been an active research area. Testing approaches are traditionally categorized as

- Black box - generating tests without knowledge of the system's implementation, or source code
- White box - generating tests derived in some way by analysis of the source code

A recurrent issue in testing is internal state: most systems contain low-level state that is not evident in higher level interfaces. Examples of this include partitioning of file data into blocks, and page tables. Maintaining this internal state often introduces bugs. Since this behavior is not visible to a black box test generator, such generators often fail to detect these bugs. Although white box test engines do have access to the code, and therefore can make use of the low-level state management code, in practice this task is difficult because the system behavior depends not only upon the input parameters, but upon the state established by all previous events. Inferring enough state properties to enable significant test improvement is extremely difficult for operating systems, as memory and storage subsystems contain much internal state. For example, although [12] has produced impressive results in essentially stateless systems, it fails to generate tests dependent on such things as the file system contents.

There are a number of approaches that have been developed to deal with state. Model-based techniques use a system model that describes the system states, providing information that can be used to verify proper behavior. A survey of model-based tools can be found in [27].

In our project, we did attempt to create a model-based test generation system. The intent was to create a test set based on the specification (i.e., model) of Caernarvon. As these tests were executed, code coverage was monitored, and the resulting coverage information was used in the generation of new tests that would target the unexecuted lines of code.

As part of this effort, we developed a new taxonomy of security flaws [34], in order to be able to sensibly describe which flaws were and weren't subject to our approach. An important foundation question also had to be resolved: in order to test our system, we had to attempt to create certain states, but the APIs used to create those states were themselves under test. As described in our paper [25], we investigated this issue and found that in practice, this was not an issue.

After designing, but before implementing our model-based test generation and feedback tool, we did an extensive experiment manually simulating its behavior using the Caernarvon file system as the target. In this experiment, we used the file system model to create tests. We executed the tests and analyzed the results. Unfortunately, as described in [35], this produced disappointing results, due to the fact that the internal file system code involved deeply complex flash memory operations, all of which resulted in enormous amounts of state that were not modeled by the file system programming interface, nor directly related to any of the file system operations. We therefore reluctantly concluded that the current model-based testing technology was not able to handle systems of the scale and complexity of Caernarvon.

6 Customers' Demand for Security is Essential

For a high-assurance product to succeed in the marketplace, there must be a qualified customer demand for a high level of security and/or privacy protection, as well as funding for the initial

upfront investment. One large potential market for the high-assurance Caernarvon operating system is the government ID market. Vendors, academics, and evaluators were attracted by initial requirements for electronic passports [7] and for government employee ID cards with better security and privacy properties [11]. In the case of federal employee ID cards, [11] specifically called for ID cards that were "secure and reliable" and "strongly resistant to identity fraud, tampering, counterfeiting, and terrorist exploitation." However, to date, actual procurements have only required Common Criteria evaluations at low and medium assurance levels (EAL4, EAL5).

Lower security products will almost always be less expensive. Unless higher assurance products are mandated, then the less secure and less expensive product will always win a competition. Until procuring agencies (both in the government and the private sector) demand strong security, the products will remain at lower assurance levels. There is also a chicken-and-egg problem here. Government standards cannot mandate products that don't even exist yet. Vendors will not risk large upfront investments to develop products without sufficient demand. An infusion of high assurance research and development funding (by governments or industry or both) could eliminate this impasse.

7 Cryptographic Library Experience

As discussed above in Section 4, the Caernarvon team developed an EAL5+ cryptographic library that Philips Semiconductors could offer to their other customers, prior to availability of the EAL7 operating system. While this introduced delay into the operating system development schedule, the cryptographic library was also needed for the operating system itself, and performing the EAL5+ evaluation gave the development team very valuable experience in how to meet Common Criteria evaluation requirements. The development team's prior experience had been with Orange Book [5] and FIPS 140 [9] evaluations, but the requirements for the Common Criteria, while similar in form, were very different in fine details.

7.1 Difficulty of Implementing DSA

One part of the cryptographic library was an implementation of the US Digital Signature Algorithm (DSA) [6]. During the code review of that module of the library, we discovered that practical software implementation of DSA was very difficult, because of subtleties in the algorithm's requirement to generate a random nonce value k that must be less than another parameter q. (The precise details of DSA are not important to this discussion.) As described in [16] and [24], if as little as one bit of k is known to an attacker, then there are practical attacks that can retrieve the secret signing key. In a later result, Bleichenbacher [4] has shown that even just a statistical bias in the randomness of k can be exploited with very large numbers of digital signatures. He pointed out that the FIPS 186-2 recommended method for choosing k was vulnerable, but this vulnerability was addresses in FIPS 186-2, Change Notice 1 [6].

When the random bits for k are generated, there is possibly an incorrect performance optimization, because the specification of DSA calls for the random bits to be less than q. Generating a random number that is less than an input parameter can be time consuming, because if the most significant bit of the random number is a one, the number could easily wind up bigger than q, requiring a time consuming generation of another random number. The

temptation is to force the most significant bit of k to always be zero, ensuring that the number will always be less than q. Unfortunately, this tempting optimization results in an exploitable vulnerability that was identified in the first code review of the Caernarvon crypto library. The details of the fix are beyond the scope of this paper. The important lesson learned was that implementing the DSA requires a highly sophisticated understanding of potential attacks on the generation of random nonces, and that simple obvious optimizations can lead to deadly vulnerabilities.

8 Smart Cards are More Complex

One lesson we quickly learned is that while smart cards are much smaller and simpler than conventional computers, that smallness can be deceiving. Making software work in such a constrained environment can be quite complex, and the security threat environment can be much more dangerous, precisely because the legitimate smart card holder may also be the attacker. Since smart cards are so portable, the attacker may be easily able to carry the card into a laboratory and use quite sophisticated technology against the card.

The impact of the tiny smart card environment has been discussed in detail in [19], and the potential for physical and side-channel attacks on smart cards is covered well by Kocher [22] and Chari, et. al. [13]. Finally, the specific Caernarvon experience with side channel attacks on hardware random number generators is discussed in [14].

9 Conclusion

Louis D. Brandeis, a former American Supreme Court Justice, once said, "There are no shortcuts to evolution."[9] We would claim that there are no shortcuts to evaluation, specifically a high assurance Common Criteria evaluation. Without a doubt, the work on the Caernarvon project was difficult and time-consuming. It required a closely linked team of collaborators - experts from multiple disciplines, multiple organizations, and, in our case, multiple countries. The team created not just the product itself, but they also created tools to automate testing, test generation, documentation generation, and vulnerability analysis. The work has been incorporated in an international standard, in large commercial server system components, and in smart card products. It earned the world's first EAL5+ Common Criteria certificate for a cryptographic library. Numerous technical papers have been published, which document the findings of the project.

The Common Criteria specification is not perfect, and, without a doubt, validating products is expensive for vendors. Despite its critics, it is the best metric we have in the area of high assurance. At high assurance levels, the Common Criteria simply mandates a discipline of good, solid, time-tested software engineering practices. Our work on the Caernarvon project has convinced us that the technical goal of an EAL7 operating system (albeit a small OS) is feasible. Its application is timely, not only in smart cards, but also in the growing area of sensors and actuators. We have demonstrated a successful model for garnering long-term executive support and project funding, which delivers interim products while maintaining a long term commitment to developing a high assurance product.

[9] http://thinkexist.com/quotation/there_are_no_shortcuts_in_evolution/213339.html

10 Acknowledgements

The Caernarvon project involved work by a number of people in addition to the authors of this paper, and we wish to acknowledge the contributions of, from IBM: Vernon Austel, Ran Canetti, Suresh Chari, Vince Diluoffo, Günter Karjoth, Gaurav Kc, Rosario Gennaro, Hugo Krawczyk, Mark Lindemann, Tal Rabin, Josyula Rao, Pankaj Rohatgi (now with Cryptography Research), Helmut Scherzer (now with Giesecke & Devrient), and Michael Steiner; from atsec: Helmut Kurth; from Philips Semiconductors (now NXP): Hans–Gerd Albertsen, Christian Brun, Ernst Haselsteiner, Stefan Kuipers, Thorwald Rabeler, and Thomas Wille; from the University of Augsberg: Wolfgang Reif, Georg Rock and Gerhard Schellhorn; from the DFKI: Axel Schairer and Werner Stephan; and from the German BSI (now seconded to the European Council): Stefan Wittmann.

References

1. *Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic requirements*, CWA 14890-1, March 2004, Comité Européen de Normalisation (CEN): Brussels, Belgium. URL: ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/eSign/cwa14890-01-2004-Mar.pdf

2. *Certification Report for Tachograph Card Version 1.0 128/64 R1.0 from ORGA Kartensysteme GmbH*, BSI-DSZ-CC-0205-2003, 22 August 2003, Bundesamt für Sicherheit in der Informationstechnik: Bonn, Germany. URL: https://www.bsi.bund.de/cae/servlet/contentblob/480764/publicationFile/30098/0205a_pdf.pdf

3. *Common Criteria for Information Technology Security Evaluation - Parts 1, 2, and 3*, Version 2.3, CCMB-2005-08-001, CCMB-2005-08-002, and CCMB-2005-08-003, 2005. URL: http://www.commoncriteriaportal.org/thecc.html

4. *Cryptologist at Lucent Technologies' Bell Labs offers improvement for future security of e-commerce: Scientist discovers significant flaw that would have threatened the integrity of on-line transactions*, 5 February 2001, Press Release: Bell Laboratories, Lucent Technologies: Murray Hill, NJ. URL: http://www.lucent.com/press/0201/010205.bla.html (downloaded 31 July 2001)

5. *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985: Washington, DC. URL: http://csrc.nist.gov/publications/history/dod85.pdf

6. *Digital Signature Standard (DSS) with Change Notice 1*, FIPS PUB 186-2, with Change Notice 1, 5 October 2001, January 2000, National Institute of Standards and Technology: Gaithersburg, MD. URL: http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2-change1.pdf

7. United States, 107th Congress, second session, 14 May 2002. Public Law No. 107-173: *Enhanced Border Security and Visa Entry Reform Act of 2002*. URL:

http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_public_laws&docid=f:publ173.107.pdf

8. *Multilevel Computer Security Requirements of the World Wide Military Command and Control System (WWMCCS)*, LCD-78-106, 5 April 1978, General Accounting Office: Washington, DC. URL: http://www.gao.gov/docdblite/summary.php?&rptno=LCD-78-106

9. *Security Requirements for Cryptographic Modules*, FIPS PUB 140-2, Change Notice 1, 10 October 2001, National Institute of Standards and Technology: Gaithersburg, MD. URL: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

10. Brand, D. *A Software Falsifier*. in **11th International Symposium on Software Reliability Engineering**. 8-11 October 2000, San Jose, CA: IEEE. p. 174-185.

11. Bush, G.W., *Policy for a Common Identification Standard for Federal Employees and Contractors*, Homeland Security Presidential Directive/Hspd-12, 27 August 2004, The White House: Washington, DC. URL: http://csrc.nist.gov/drivers/documents/Presidential-Directive-Hspd-12.html

12. Cadar, C. and D. Engler. *Execution Generated Test Cases: How to Make Systems Code Crash Itself*. in **Proceedings of the 12th International SPIN Workshop on Model Checking of Software**. August 2005. URL: http://citeseer.ist.psu.edu/cadar05execution.html

13. Chari, S., J.R. Rao, and P. Rohatgi. *Template Attacks*. in **Cryptographic Hardware and Embedded Systems - CHES 2002**. 2002, Redwood Shores, CA:Lecture Notes in Computer Science Vol. 2523. Springer Verlag. p. 13-28.

14. Chari, S.N., V.V. Diluoffo, P.A. Karger, E.R. Palmer, T. Rabin, J.R. Rao, P. Rohatgi, H. Scherzer, M. Steiner, and D.C. Toll. *Designing a Side Channel Resistant Random Number Generator*. in **Smart Card Research and Advanced Application: 9th IFIP WG8.8/11.2 International Conference: CARDIS 2010**. 14-16 April 2010, Passau, Germany:Lecture Notes in Computer Science Vol. 6035. Springer. p. 49-64.

15. Dijkstra, E.W., *The Structure of the "THE"-Multiprogramming System.* **Comm. ACM**, May 1968. **11**(5): p. 341-346.

16. Howgrave-Graham, N.A. and N.P. Smart, *Lattice Attacks on Digital Signature Schemes.* **Designs, Codes and Cryptography**, 2001. **23**: p. 283-290.

17. Hutter, D., H. Mantel, G. Rock, W. Stephan, A. Wolpers, M. Balser, W. Reif, G. Schellhorn, and K. Stenzel. *VSE: Controlling the Complexity in Formal Software Developments*. in **Applied Formal Methods - FM-Trends 98**. 7-9 October 1998, Boppard, Germany:Lecture Notes in Computer Science Vol. 1641. Springer. p. 351-358. URL: http://www.dfki.de/vse/papers/hmrs98.ps.gz

18. Karger, P.A. and H. Kurth. *Increased Information Flow Needs for High-Assurance Composite Evaluations*. in **Second IEEE International Information Assurance Workshop**. 8-9 April 2004, Charlotte, NC: IEEE Computer Society. p. 129-140.

19. Karger, P.A., D.C. Toll, E.R. Palmer, S.K. McIntosh, S.M. Weber, and J. Edwards. *Implementing a High-Assurance Smart-Card OS*. in **Financial Cryptography and Data Security '10**. 25-28 January 2010, Tenerife, Spain:Lecture Notes in Computer Science Vol. 6052. Springer. p. 51-65.

20. Karger, P.A., M.E. Zurko, D.W. Bonin, A.H. Mason, and C.E. Kahn, *A Retrospective on the VAX VMM Security Kernel.* **IEEE Transactions on Software Engineering**, November 1991. **17**(11): p. 1147-1165.

21. Klein, G., K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. *seL4: Formal Verification of an OS Kernel*. in **Proceedings of the 22nd ACM Symposium on Operating Systems Principles**. October 2009, Big Sky, MT: ACM. URL: http://ertos.nicta.com.au/publications/papers/Klein_EHACDEEKNSTW_09.pdf

22. Kocher, P., J. Jaffe, and B. Jun. *Differential Power Analysis: Leaking Secrets*. in **Proceedings of Crypto '99**. August 1999, Santa Barbara, CA:Lecture Notes in Computer Science Vol. 1666. Springer Verlag. p. 388-397.

23. Mersey, D., *Caernarvon Castle - Fit for a Prince*, 2007. URL: http://www.castlewales.com/caernarf.html

24. Nguyen, P.Q. and I.E. Shparlinski, *The Insecurity of the Digital Signature Algorithm with Partially Known Nonces.* **Journal of Cryptography**, March 2008. **15**(3): p. 151-176.

25. Paradkar, A., S. McIntosh, S. Weber, D. Toll, P. Karger, and M. Kaplan. *Chicken & Egg: Dependencies in Security Testing and Compliance with Common Criteria Evaluations*. in **IEEE International Symposium on Secure Software Engineering (ISSSE '06)**. 13-15 March 2006, Arlington, VA: IEEE Computer Society. p. 65-74.

26. Parnas, D.L., *On the Criteria to be Used in Decomposing Systems into Modules.* **Comm. ACM**, December 1972. **15**(12): p. 1053-1058.

27. Redwine, S., *Introduction to Modeling Tools for Software Security*, 21 February 2007, National Cyber Security Division, U.S. Department of Homeland Security. URL: https://buildsecurityin.us-cert.gov/daisy/bsi/articles/tools/modeling/698.html?branch=1&language=1

28. Schell, R.R. *Computer security: the Achilles' heel of the electronic Air Force?* in **Air University Review**. January-February 1979, Vol. 30. p. 16-33. URL: http://www.airpower.maxwell.af.mil/airchronicles/aureview/1979/jan-feb/schell.html

29. Schell, R.R., *A Security Kernel for a Multiprocessor Microcomputer*. **Computer**, July 1983. **16**(7): p. 47-53.

30. Schellhorn, G., W. Reif, A. Schairer, P. Karger, V. Austel, and D. Toll. *Verification of a Formal Security Model for Multiapplicative Smart Cards*. in **6th European Symposium on Research in Computer Security (ESORICS 2000)**. 4-6 October 2000, Toulouse, France:Lecture Notes in Computer Science Vol. 1895. Springer-Verlag. p. 17-36.

31. Scherzer, H., R. Canetti, P.A. Karger, H. Krawczyk, T. Rabin, and D.C. Toll. *Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card*. in **8th European Symposium on Research in Computer Security (ESORICS 2003)**. 13-15 October 2003, Gjøvik, Norway:Lecture Notes in Computer Science Vol. 2808. Springer Verlag. p. 181-200.

32. Toll, D.C., P.A. Karger, E.R. Palmer, S.K. McIntosh, and S. Weber, *The Caernarvon Secure Embedded Operating System*. **Operating Systems Review**, January 2008. **42**(1): p. 32-39.

33. Toll, D.C., S. Weber, P.A. Karger, E.R. Palmer, and S.K. McIntosh. *Tooling in Support of Common Criteria Evaluation of a High Assurance Operating System*. in **Build Security In**. 3 April 2008, Department of Homeland Security. URL: https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/lessons/961.html

34. Weber, S., P.A. Karger, and A. Paradkar, *A Software Flaw Taxonomy: Aiming Tools At Security*, RC 23538 (W0502-133), 25 February 2005, IBM Research Division, Thomas J. Watson Research Center: Yorktown Heights, NY. URL: http://www.research.ibm.com/resources/paper_search.html

35. Weber, S., S.K. McIntosh, A. Paradkar, D.C. Toll, P.A. Karger, M. Kaplan, and E.R. Palmer. *The Feasibility of Automated Feedback-Directed Test Generation: A Case Study of a High-Assurance Operating System*. in **19th International Symposium on Software Reliability Engineering**. 11-14 November 2008, Redmond, WA: IEEE Computer Society. p. 229-238.