

Secure Software and Systems: Follow Five Principles and Prove It

To be truly secure, computer systems, their operating systems and software must meet stringent criteria—criteria not determined by the market but by expert testing up to and including the National Security Administration.

by David Kleidermacher
Green Hills Software, Inc.

The world has become accustomed to the fail-first, patch-later mentality of insecure software and computing infrastructure. As Michael Vatis, a former director of the FBI's National Infrastructure Protection Center, has said: "The vulnerabilities are endemic because we have whole networks and infrastructures built on software that's insecure. Any given day, some new vulnerability pops up." Thus, much of the world's critical infrastructure, financial networks, medical information systems, telecommunications gear and portable mobile devices are open to compromise by determined individuals, corporations, organized crime and nation states.

While the world spends untold billions on snake-oil solutions—firewalls, filters and Patch Tuesdays—security experts know full well that the only true path to security is through high assurance. Users of high assurance software can have high confidence in the ability of that software to securely perform its intended function. Almost all of the world's commercial software is low assurance, and our confidence level in its security matches. Yet recent developments have proven that high assurance software can be practical, even for complex applications such as operating systems.

Green Hills has developed and applied an architecture—the Secure Separation Architecture—to create a wide variety of highly secure components, applications and complete systems. Such an architecture should help computer and security professionals as well as the average consumer understand what is meant by high assurance. The architecture prescribes a set of five principles to be

used in the creation of secure software and systems. Some of these may be familiar, while others may not be obvious; all are critical:

1. Minimal Implementation

It is much harder to create simple, elegant solutions to problems than complex, convoluted ones. Most software developers do not work in an environment in which producing the absolute minimal possible solution to a problem is an unwavering requirement. Spaghetti code is the source of vulnerabilities that run rampant in software and provide the avenue of exploitation for hackers.

2. Componentization

Compose large software systems from small components, each of which is easily maintained by (ideally) a single engineer who understands every single line of code. Use well-defined, documented interfaces between components. An important corollary to this rule is that security-enforcing functionality should be placed into separate components, so that security functionality is protected from compromise by unrelated functionality. Componentization provides many benefits, including improved testability, auditability, data isolation and damage limitation. Componentization can prevent a failure in one component from devolving into a system failure.

3. Least Privilege

Components must be given access to only those resources (e.g. communication pathways, I/O devices, system services, information) that are required. Access control must be mandatory

Name	Title	Security Level	Threat Environment
SKPP	Separation Kernel in High Robustness Environments	EAL 6+	“management of classified and other high-valued information, whose confidentiality, integrity or releasability must be protected” “presence of both sophisticated threat agents and high-value resources”
CAPP	Controlled Access Protection Profile	EAL 4+	“non-hostile and well-managed user community” “inadvertent or casual attempts to breach the system security” “not intended to be applicable to circumstances in which protection is required against determined attempts by hostile and well-funded attackers”
LSPP	Labeled Security Protection Profile	EAL 4+	“non-hostile and well-managed user community” “inadvertent or casual attempts to breach the system security” “not intended to be applicable to circumstances in which protection is required against determined attempts by hostile and well-funded attackers”
SLOS	Single Level Operating Systems in Medium Robustness Environments	EAL 4+	“low value data” or “closed environment” Not appropriate for “organization’s most sensitive/proprietary information” when exposed to “a publicly accessible network” “motivation of the threat agents will be average”
MLOS	Multilevel Operating Systems in Medium Robustness Environments	EAL 4+	“low value data” or “closed environment” Not appropriate for “organization’s most sensitive/proprietary information” when exposed to “a publicly accessible network” “motivation of the threat agents will be average”

Table 1 Operating System Security Evaluation Standards

for critical system resources and information. This is a biggy. In most operating systems, it is the birthright of any program to access the file system, launch other programs, and manipulate system devices. This lack of least privilege design enables a simple vulnerability, such as a buffer overflow, to be exploited to maximum deleterious effect.

4. Secure Development Process

For the critical security-enforcing components, the software development process must meet the highest levels of reliability assurance, such as DO-178B Level A (a standard for ensuring the safety of flight-critical systems in commercial aircraft) or equivalent—as well as the highest levels of security assurance—EAL 6/7 or equivalent. This assurance process will cover numerous controls, including configuration management, coding standards, testing, formal design, formal proof of security policy, etc.

5. Independent Expert Validation

Intrinsic assurance evidence must be evaluated and confirmed by independent experts. Consumers who read about a vendor’s claim of “certifiable” should interpret such hyperbole as “not certified.” For example, to achieve certification to the high assurance Separation Kernel Protection Profile (SKPP) in the United States, a separation kernel must not only have all of its design documentation, testing and formal methods painstakingly evaluated by an accredited Common Criteria testing lab, but the product must also undergo penetration testing by the NSA’s penetration testing experts who have complete access to the source code and essentially unlimited resources with which to craft attack vectors.

Software that fulfills all five principles (it’s easy to see how the lack of any one of them can be fatal) can be trusted to manage and protect high-value assets, even if those assets are under attack by the most determined and resourceful enemies (read “exposed to the Internet”). The Secure Separation Architecture

does not attempt to rigorously define “secure development process” or specifically how the principles of least privilege apply to a particular software component or system. Interpretations will necessarily vary depending on the situation.

Common Criteria

Common Criteria is an important part of the answer to this interpretation dilemma. Common Criteria—more formally known as ISO/IEC 15408—is the international standard for evaluating the security of IT systems. Under Common Criteria, IT products are evaluated against Protection Profiles, which specify the product family’s security functional requirements and security level called the Evaluated Assurance Level, or EAL. For example, there are protection profiles for firewalls, antivirus applications and operating systems.

Protection profiles themselves must be evaluated, ensuring that products are measured against a well-understood, valid and accepted standard. There are five operating system-related protection profiles that have been evaluated by the U.S. Government (<http://www.niap-ccevs.org/cc-scheme/pp/>). These operating system specifications have varying security levels, making them more or less appropriate for certain threat environments. The operating system profiles and their applicable threat environments are listed in Table 1.

As can be seen from Table 1, only the SKPP is appropriate to protect high-value information exposed to the threat of sophisticated and determined attackers. In agreement with the secure development process principle, according to Common Criteria, EAL 4 “is the highest level at which it is likely to be economically feasible to retrofit an existing product line.” There is more to security than using the word “secure” or “trusted” in product names and press releases.

Defense Against the Dark Arts

In 2007, IEEE Spectrum published *The Athens Affair* (<http://www.spectrum.ieee.org/jul07/5280>), detailing the recent incident in which the Greek Prime Minister and a slew of other high ranking dignitaries had their mobile devices bugged. Hackers infiltrated the cellular provider's computer systems to inject software that maliciously redirected private communications. Investigators concluded that the computers were "reprogrammed with a finesse and sophistication rarely seen before or since." *The Athens Affair* was a well-funded and determined breach of system security by hostile entities.

The number of detected intrusions into U.S. government networks has been steadily on the rise. For example, the Department of Homeland Security suffered 850 known cyber attacks in a two-year period beginning in 2005. Security experts believe that the actual number of successful breaches is far higher. The truth is that we have no idea what has already been stolen or commandeered, awaiting the opportune moment to strike. Too many IT professionals believe that encrypted communications, Web filters, firewalls and other security appliances are sufficient to protect valuable assets. Talk to one of the leading ethical hacking firms (better yet, hire them to attempt to infiltrate your corporate network), and you will soon be convinced that these technologies are akin to putting a padlock on a screen door. Ironically, the security appliances themselves run insecure operating systems.

Many of the security problems that plague IT infrastructure can be cured with a Secure Separation Architecture and its use of

high assurance certified products, such as SKPP-certified operating systems. The requirements of the SKPP are far more stringent than any other operating system security standard. The resulting assurance, or confidence, that developers, users and other stakeholders are therefore able to derive from this certification is extremely high and unprecedented in the world of computer security. The Secure Separation Architecture is now being applied to solve important security problems in corporate, civil and government computing infrastructure. ■

Green Hills Software, Inc.
Santa Barbara, CA.
(805) 965-6044.
[www.ghs.com].