

Four Ways to Improve Security

Many of today's commercial IT security products aren't as secure as they need to be. In fact, it's been this way from the beginning of the emergence of security products.

Yet, processes and techniques ranging from the near horizon to

today's threats against security systems, including buffer overflows.

Customers should verify that vendors have a QC process in place, one that the customer, or someone competent on his or her staff, recognizes. I mention two not as recommendations, but merely as examples. Carnegie Mellon's Software Engineering Institute (SEI) has a capability maturity model (CMM; www.sei.cmu.edu/cmm/cmm.html) with levels 1 through 5. Vendors' claims of being SEI CMM level 5 compliant carry a lot of weight. Another option is Watts Humphrey's Team Software Process and Personal Software Process (www.sei.cmu.edu/tsp), which have impressive statistics on reducing the number of errors per thousand lines of code. Stay away from vendors claiming to produce quality software with home-grown QC programs—their claims are worth less than the hot air used to propound them. Look for third-party QC attestations.

Most QC regimes assume a benign environment; but a security product must survive in a malicious one. Thus, we need additional high-value QC features in the security domain: thorough *bounds checking* (inputs are of the expected type: if numeric, in the expected range; if character strings, the length doesn't exceed the internal buffer size), and *input scrubbing* (reasonableness tests: if an input should be a single word of text, a character string containing multiple words is wrong, even if it fits in the buffer.) will knock out the vast majority of common security flaws; do them first. Next, there should be strong *configuration control*, during the design process and in the

BRIAN SNOW
US National
Security
Agency

over the horizon (still a gleam in a researcher's eye) could dramatically improve the situation. To get there, customers must demand more from vendors today to get the security they'll need in the products they'll buy tomorrow.

How can you tell if a security product (or a product that includes security components) can secure your application? How can you be certain that a product will fully deliver on its claims that it will protect against malice in a deployed environment? Unfortunately, few vendors—and even fewer customers—can make these judgments. This article won't make you a security wizard, but it will give you a feel for what to look for in, and when to be concerned about, a vendor's claims.

To ensure that a product has a chance of being secure, customers should check that vendors use adequate approaches in four primary areas. In order of importance (and of their maturity and availability), they are

- quality-control (QC) mechanisms,
- cryptographic primitives,
- hardware assist mechanisms, and
- separation mechanisms.

When none of these factors is

present, the product will provide the appearance of security—but not the reality. Security product vendors might address the most egregious and visible attacks, such as buffer overflows, but until they address each of these four points, many vulnerabilities will remain in their products.

Quality control

Whether a product is software, hardware, or a combination, it's critical that it fully and accurately implements its specifications with as few flaws as possible—something that isn't possible without using QC. In general, hardware development is a more mature process than software development, so I'll primarily focus on software in this discussion. But my comments are broad enough to apply to both domains.

QC is an essential primitive for the future of security; without it, we simply can't improve products and succeed against threats. E-commerce and, in particular, digital rights management are current drivers that will help us maintain a strong focus on quality—along with the increasing tempo of lawsuits. Today's buffer overflow vulnerability is an embarrassment and a highly visible symptom of the lack of QC in current products. One single step—robust QC—can remove the bulk of

product as delivered, to verify configuration compliance throughout its deployed life. If present, cryptographic functions can help enforce configuration control by providing digital signature checks of critical modules prior to each execution and ensure that there have been no malicious or benign modifications.

A product design process should enforce modularity, providing clean abstract module interfaces and independent reviews for each module for specification compliance. Thus it's inappropriate for a designer to review his or her own work; whatever blind spot the designer had while building it will still be present during its review. Independent review is critical for security products—they make it much more difficult for a malicious opponent to corrupt the design process (or designer).

Not only is QC a key first step toward a secure product, it arguably provides the most bang for the buck.

Cryptographic primitives

Cryptography is often required in security products and, if present, must be robust. This is a knowledge intensive domain where minor misunderstandings can have catastrophic consequences, so much care is needed. Fortunately, it's easy to specify some primary pitfalls.

First, avoid proprietary processes, and use known, well-vetted processes and protocols. The US National Institute of Standards and Technology's cryptographic toolkit is a suite of cryptographic primitives the Advanced Encryption Standard (AES) with appropriate modes, the Secure Hash Algorithm (SHA-1), Digital Signature Algorithm, elliptic curve suites, and, soon, key management protocols (<http://csrc.nist.gov/CryptoToolkit>). The financial sector offers a suite of cryptographic standards through American National Standard Institute X9 efforts (http://webstore.ansi.org/ansidocstore/dept.asp?dept_id=80). Even if

a process or protocol is well-known, use only those endorsed by large well-established standards bodies. Participants at the Crypto 2004 rump session (www.iacr.org/conferences/crypto2004) broke essentially all publicly known hash codes except SHA-1; and even SHA-1 might have recently discovered weaknesses (<http://www.pcworld.com/news/article/0,aid,119726,00.asp>)—ample evidence that using well-vetted, widely studied, and formally endorsed cryptographic primitives provides your best chance at strong security.

The implementation is of equal or greater concern: when possible, use well-known reference implementations or robust hardware implementations that provide high-level, well-specified, and thoroughly studied interfaces and function calls. Developing a new proprietary implementation of a widely endorsed algorithm should be a last resort. Pursue it only when you can't meet a product's specification in any other way. Also, be wary of proprietary or unique modes of operation; well-specified modes are available that cover a huge spectrum of needs. Stick with them.

It might sound like I'm trying to constrain the emergence of new cryptography approaches; far from it. Academic and corporate research units are aggressively looking for new primitives and this can be exciting and important work. But wait until standards bodies recognize and endorse some of their efforts. Trying out new cryptographic processes on customers before the community analyzes and reports on them doesn't make good security sense.

Hardware assist mechanisms

To improve performance, designers sometimes implement cryptographic functions in hardware, either in stand-alone devices such as link encryptors, cryptographic coprocessors, point-of-sale smart cards, or

Universal Serial Bus I/O port or FireWire bus dongles.

Although generally more expensive than software-only approaches, carefully apportioned hardware functionality architected from a systems' viewpoint can provide significant gains in security as well as performance. This approach lets cryptographic keys and functions remain inside a trusted device, protected from external threats. Software and hardware components can practice *mutual suspicion*, each checking the other and alerting the system to malfunction or malice in the other component.

The additional security provided by encapsulating critical functions in hardware can be significant because of strong potential synergy. More information is available in the literature, such as the proceedings of the Cryptographic Hardware and Embedded Systems conferences (<http://islab.oregonstate.edu/ches>).

Today, many vendors see the security value that properly designed hardware components can provide for security products. For example, the Trusted Computing Group (www.trustedcomputinggroup.org/home) is a vendor consortium providing hardware assistance for a variety of security functions.

Although the additional costs of hardware-assisted mechanisms might limit a broad market penetration, security gains can be significant for customers that need it, such as banks and large corporations with high-value intellectual property.

Separation mechanisms

One reason security is so difficult to implement in today's IT environment is a resource-sharing design paradigm underlying computers and other components. In the early days of computers, memory was expensive, so different processes shared it as much as possible, a practice that continues today. If your platform shares resources, it's difficult to impose ro-

bust separation in that environment; and separation is the best one-word synopsis for security functionality, even if it is an oversimplification.

When you install a separation mechanism on a share-based platform, residual covert channels remain (see www.fas.org/irp/nsa/rainbow/tg030.htm). Academics have studied covert channels since at least the '70s, but vendors have paid no attention to the problem in today's security products. Why? Because attackers succeed with simple mechanisms made possible by poor QC.

When the quality of security products improves enough to remove the low-hanging fruit (easy vulnerabilities) that attackers (at least those into electronic theft) exploit today, they will then pay more attention to covert-channel mechanisms. Why? Because a covert-channel attack is about as difficult to implement as the ones attackers already are using. Today's IT design paradigms make covert-channel defense very difficult.

If you want robust security against current and future attacks, there's hope. Researchers are working on separation kernels that let IT systems partition memory and support information flow restrictions for applications that use them. For example, Green Hills Software's work on a separation kernel for flight safety (www.afei.org/brochure/4af3/presentations/mark_Van_Fleet.pdf) might also generalize to kernel structures underlying commercial operating systems (www.ghs.com/news/20041129_INTEGRITY_PC.html). If that were to happen, storage channels might be eliminated and timing channels reduced by multiple orders of magnitude—a stunning and significant security milestone.

These strong separation mechanisms would position security products not only to perform better today, but would also allow them to be one step ahead by removing one of the next obvious attack paths enemies might take.

As the security industry matures, I look forward to more robust QC regimes that vendors could adopt for initial costs lower than today's. I expect vendors to offer suites of interoperable products, some of which would provide higher assurance than others, even though they interoperate using the same security functions. Customers could purchase products at price points that match their risk environments: inexpensive software-only products for home use and mom-and-pop operations; more robust software and hardware blends for customers with more at risk. For this, we'd need quality-of-implementation negotiation protocols and appropriate modularity and partitioning of functionality so a module could exist either in hardware or software as the situation demanded.

There is still much work ahead in providing effective tamper detection or resistance in security products and correctly identifying network attack sources. Formal methods are mature enough that adopting them on a broader scale could bring huge improvements in product quality and the provability of certain security properties. We also can find and apply other highly desirable security design factors from the safety, reliability, and maintainability disciplines. □

Brian Snow is technical director for the US National Security Agency's Associate Directorate for Education and Training. In his 33-year career at NSA in research and information security, he's always been interested in anything that improves his customers' security. He has an MA in mathematics from the University of Colorado. Contact him at bdsnow@nsa.gov.