Sponsored by

Enterprise Security Products

Application Security

Ensuring application security across an enterprise requires a holistic approach to software development and procurement. Security risks must be addressed at the application code level, and techniques and best practices should focus on detecting and eliminating software security vulnerabilities. IT departments that practice comprehensive application security protect their organizations from threats and unauthorized access that could lead to data theft.

Yet application security is becoming harder and harder to achieve as threats grow more pervasive and protecting software fails to gain the attention it deserves in the enterprise. In this eGuide, *CSO* and its sister publications *Network World*, *Computerworld*, and *InfoWorld* present a collection of articles that examine the current threats to application security and what companies can do to protect themselves.

In this eGuide

Security News

Secunia Offers to Coordinate Vulnerability Disclosure on Behalf of Researchers

By Lucian Constantin • IDG News Service

New vulnerability coordination program aims to reward security researchers and make their job easier

Danish vulnerability management company Secunia aims to make the task of reporting software vulnerabilities easier for security researchers by offering to coordinate disclosure with vendors on their behalf.

The Secunia Vulnerability Coordination Reward Programme (SVCRP) is the latest addition to a list of offerings like TippingPoint's Zero Day Initiative or Verisign's iDefense Labs Vulnerability Contributor Program, which allow researchers to avoid the hassle of dealing with different vendor bug reporting policies.

However, according to Carsten Eiram, Secunia's chief security specialist, SVCRP doesn't aim to be an alternative to these programs, but to complement them.

"Other major vulnerability coordination offerings exist but most have a business model wrapped around them," Eiram said. "Most other schemes pay researchers for their discoveries, and, while these offerings are excellent for researchers, the companies are, naturally, very selective in which vulnerabilities they wish to purchase and coordinate."

Secunia plans to fill the void left by other programs by accepting the vulnerabilities they reject, regardless of their classification and as long as they are in off-the-shelf products. Flaws discovered in online services such as Facebook, for example, do not qualify.

The company won't profit directly from SVCRP and doesn't plan to provide advance notification about the reported flaws to its customers, as other companies do. "All customers, as well as the community at large, will receive the information simultaneously when the Secunia advisory is published," the firm said.

Researchers will continue to receive payments they are entitled to from vendors for disclosing vulnerabilities even if they use SVCRP for coordination, Secunia said. However, vendors have the final word on whether they will pay out rewards to researchers who offload vulnerability coordination work to companies such as Secunia.

For example, the guidelines of Google's Chromium Vulnerability Rewards Program make it clear that vulnerabilities disclosed through brokers and other third parties are not likely to receive a bounty.

"I do not know whether Google would pay for vulnerabilities coordinated via Secunia, however, considering the nature and the purpose of our programme, especially

Focal point

Unlike other programs, SVCRP doesn't require researchers to provide working exploits for the vulnerabilities they find. This allows security researchers to focus more on what they do best: finding vulnerabilities.

the fact that we don't have a business model wrapped around it, seems to make it more likely that Google would accept reports via Secunia rather than other programmes," said Thomas Kristensen, Secunia's chief security officer and co-founder.

Secunia's own experts will investigate and confirm every submitted vulnerability before sharing it with the corresponding vendor, so researchers will get independent validation of their findings, and affected companies will receive consistent reports.

Unlike other programs, SVCRP doesn't require researchers to provide working exploits for the vulnerabilities they find. Kristensen said that providing only information that allows his company to reproduce the findings will suffice, although additional details are welcome.

This allows security researchers to focus more on what they do best—finding vulnerabilities—than on writing reliable exploits, which can take a considerable amount of time. Not all researchers are skilled exploit writers and not every exploit writer is necessarily good at finding vulnerabilities.

Secunia will also reward the program's most valuable contributors, but not with cash. Instead, each year the company will invite to a top security conference the researcher who reported the most interesting vulnerability and the one who submitted the best-quality reports throughout the year.

The Secunia chief security officer would like to see more software companies setting up vulnerability bounty programs in the future. "Researchers do a lot of free Quality Assurance work for vendors. Any reward for their work and their willingness to coordinate the disclosure would be a good move from the vendors," Kristensen said.

Unfortunately, many companies don't yet see the value of that and are not interested in encouraging outsiders to point out their software's flaws, he said.

IDG News Service is an affiliate of CSO.

Expert Advice

Software Security Basics for Application Development Managers

By Mark S. Merkow and Lakshmikanth Raghavan • CSO

Fewer security holes mean better software quality and lower costs.

If we have learned nothing else from sixty-plus years of software development it's that secure, high-quality code does not happen by accident. National governments have long understood that quality and security must be specified, designed, and implemented from soup-to-nuts throughout the software development lifecycle (SDLC).

Governments go to great length to assure that only provably secure software is used to protect national secrets. Unfortunately, this philosophy never spilled over into the commercial world and what we're left with today is a precarious branch filled with brittle applications of questionable reliability.

This article address these issues and offers solutions to help you avoid repeating the same mistakes that lead to even more insecure and unreliable software and systems. We'll look at what people expect when they acquire software and what they actually get, reasons for the gaps, what you as managers can do to improve your own organization's software development practices and measure the progress of your success.

What People Expect in Software

Software is useful for what it does. People purchase software because it fulfills their need to perform some function. These functions (or features) can be as simple as allowing a user to type a letter or as complex as calculating the fuel consumption for a rocket trip to the moon. Functions and features are the reasons people purchase or pay for the development of software and it's in these terms that people think about software.

People also (erroneously) assume that the software they purchase is written with some degree of quality. When you purchase a software package you just assume it will operate as advertised and you never really think about how well the program does its job—just as long as it works!

The sad truth is that most software is flawed straight out of the box and these flaws can threaten the security and safety of the very systems on which they operate. These flaws are not just present in the traditional computers we use everyday, but also in critical devices, like our cell phones and medical devices—think pacemakers and cars—and national infrastructures like banking and finance, energy, and telecommunications.

Programmers are taught to write code—they are not taught how to write good code.

The good, the bad, and the vulnerable

Programmers are taught to write code—they are not taught how to write good code.

Organizations incent programmers to write more code, leaving good code out of the equation while cheap code and rapidly-developed code dominates the landscape.

Web applications especially are inherently flawed with certain types of vulnerabilities (e.g. Cross Site Scripting) unless the developer has made a conscious effort to prevent it. If the developer fails to include appropriate output encoding routines and input validation routines, the application will most certainly be vulnerable by default to XSS.

To a developer, the software may in fact work just as he intended it to work but never tested it to see how it behaves when it's being fed malicious input or is under direct attack.

Writing software, like driving a car, is a habit. Until someone teaches us how to drive safely, we don't personally know the dangers of driving and the skills needed to prevent or avoid accidents. Cars often have safety mechanisms built into them, but as drivers, we have to consciously use our own safe driving skills. Experience teaches us that we are better off instilling safe driving skills before we let people loose on the roads, since their first accident may be their last.

How Bad is the Problem, Really?

In 2008 there was significant increase in the count of identified vulnerabilities in commercial software—a 13.5 percent increase compared to 2007. The overall severity of vulnerabilities also increased, with high and critical severity vulnerabilities up 15.3 percent. Medium severity vulnerabilities were up 67.5 percent and nearly 92 percent of 2008 vulnerabilities reported can be exploited remotely.

Of all the vulnerabilities disclosed in 2008, only 47 percent are able to be corrected through vendor patches. At the end of 2008, 74% of all web application vulnerabilities disclosed in 2008 had no available patch to fix them.

Web applications are the Achilles Heel of corporate IT security too. Every year organizations across the globe spend millions of dollars on securing their software infrastructure—a significant portion of an entire year's IT budget. Software security spending tends to focus on detecting existing vulnerabilities in the software organizations already own and finding ways to reduce the associated risks of using it. Rewriting software, whether to fix a problem or fundamentally change what the software does, also results in tremendous corporate expenditures year after year. Bad code also negatively affects productivity whenever the application or a system goes down, leading to indirect or direct losses to the business. Other web application flaws lead to brand damage, reputation damage, information theft, and denial of service problems.

Don't Bolt Security On—Build It In!

Software flaws appear in software because somewhere along the specification, development, and testing conveyor

belt, requirements that mandate secure software fell on the floor and were neglected. Software is only secure when it's designed for security—most attempts at bolting on security after the fact yield even more problems than when it's considered from the beginning of a development effort.

To actually move the needle on progress for software security requires that security be built in to the development life cycle itself and begins with a management-sponsored Secure Coding Initiative to fundamentally improve how an organization thinks about and develops software for public use, for in-house use, and for sale to others.

Any secure coding initiative must deal with all stages of a program's lifecycle. Secure programs are secure by design, during development, and by default. As with any significant change initiative, education plays a key role. Education is the cornerstone of any effective intervention to improve software quality and should be treated as non-optional. Prior to relying on the people involved in software development for secure code, it's essential that training in secure design and coding is administered to help the analysts, designers, and developers to better understand how incomplete analysis, poor design decisions, and poor coding practices contribute to application and system vulnerabilities.

Once educated, SDLC participants begin behaving differently and start to serve as an internal checks-and-balances mechanism that catches problems earlier in the SDLC and leads to lower costs of development and higher quality systems.

Education needs to prepare personnel in the basics of Web application flaws, familiarize them with the hacking tools of the trade intended to break application software, and prepare them to carry out their craft with new skills and techniques.

From the earliest days of software development, studies have shown that the cost of remediating vulnerabilities or flaws in design are far lower when they're caught and fixed during the early requirements/design phases than they are after launching the software into production. Therefore, the earlier you integrate security processes with the development life cycle, the cheaper software development becomes in the long haul.

These security processes are often just "common sense" improvements and any organization can and should adopt them into their existing environment. There is no one right way to implement these processes—each organization will have to fine tune and customize them for their specific development and operating environments. These process improvements add more accountability and structure into the system, too. There are a number of well-accepted secure software development methodologies, including Microsoft's Secure Development Lifecycle (SDL), Cigital's Touchpoints, and the one we'll examine here, called CLASP.

Comprehensive, Lightweight Application Security Process

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. One of their more prominent projects is called the Comprehensive, Lightweight Application Security Process, or CLASP.

CLASP is a pre-defined set of documented processes and tools that can be integrated into any software development process. It is designed to be both easy to adopt and effective.

CLASP uses a prescriptive approach, documenting activities that organizations should be doing. CLASP is rich with an extensive collection of freely available and open-source security resources that make implementing those activities practical and achievable.

Think of CLASP as a resource library to avoid re-invent-

ing the wheel when you come across the need for new processes or new ideas for secure software development.

You can obtain a free copy of CLASP in book form or from the OWASP CLASP Wiki.

CLASP Highlights

CLASP provides extensive detailed information on:

- Concepts behind CLASP to get started
- Seven key best practices that define CLASP
- High-level security services that serve as a foundation
- Core security principles for software development
- Abstract roles that are typically involved in software development
- Activities to augment the development process to build more secure software
- CLASP process engineering and roadmaps
- Coding guidelines to help developers and auditors when reviewing code
- A lexicon of vulnerabilities that occur in source code
- A searchable vulnerability checklist in Excel format for the CLASP vulnerability lexicon

Implementing software application security best practices requires a reliable process to guide a development team in creating and deploying an application that is as resistant as possible to security attacks. Within a software development project, the CLASP best practices are the basis of all security-related software development activities. Here are the seven CLASP best practices:

- **Best Practice 1: Institute awareness programs**
- **Best Practice 2: Perform application assessments**
- **Best Practice 3: Capture security requirements**
- **Best Practice 4: Implement secure development practices**
- **Best Practice 5: Build vulnerability remediation procedures**
- **Best Practice 6: Define and monitor metrics**
- **Best Practice 7: Publish operational security guidelines**

Essential security concepts and techniques may be foreign to your organization's software developers and others involved in application development and deployment. So it is imperative at the outset to educate everyone involved. Awareness programs can be readily implemented, using external expert resources, as appropriate, and deliver a high return by helping to ensure that other activities promoting secure software will be implemented effectively. Here are some tips to help with security awareness:

Provide security training to all team members

Before team members can reasonably be held accountable for security issues, you must ensure they have had adequate exposure to those issues. This is best done with a training program. Everyone on the team should receive training introducing them to basic security concepts and secure development process that is used within the organization.

Promote awareness of the local security setting

Everyone on a development project should be familiar with the security requirements of the system, including the basic threat model. When such documents are produced, they should be distributed and presented to team members, and you should solicit and encourage feedback from all parties on the team.

Institute accountability for security issues

Traditional accountability within development organiza-

tions is based primarily on schedule and quality. Security should be treated in much the same way as any other quality consideration.

Appoint a project security officer

An excellent way to increase security awareness throughout the development lifecycle is to designate a team member as the project security officer, particularly someone who is enthusiastic about security.

Institute rewards for handling of security issues

Accountability is a necessity for raising security awareness, but another highly effective way is to institute reward programs for doing a job well done with regard to security. For example, it is recommended to reward someone for following security guidelines consistently over a period of time—particularly if the result is that no incidents are associated with that person.

Testing and assessment functions are typically owned by a test analyst or by the QA organization but can span the entire SDLC. CLASP offers detailed guidance for each of the following areas of application assessments:

- Identify, implement, and perform security tests
- Find security problems not found by implementation review
- Find security risks introduced by the operational environment
- Act as a defense-in-depth mechanism, catching failures in design, specification, or implementation
- Perform security analysis of system requirements and design (threat modeling)
- Assess likely system risks in a timely and cost-effective manner by analyzing the requirements and design
- Identify high-level system threats that are documented neither in requirements nor in supplemental documentation
- Identify inadequate or improper security requirements
- Assess the security impact of non-security requirements
- Perform source-level security review
- Find security vulnerabilities introduced into implementation
- Research and assess security posture of technology solutions
- Assess security risks in third-party components
- Determine how effective a technology is likely to be at alleviating risks
- Verify security attributes of resources
- Confirm that software abides by previously defined security policies

It is especially important in the context of application updates and enhancements to define which steps will be taken to identify, assess, prioritize, and remediate vulnerabilities. CLASP guidance can be found for:

- Addressing reported security issues
- Ensuring that identified security risks in an implementation are properly considered
- Managing the security issue disclosure process
- Communicating effectively with outside security researchers when security issues are identified in released software, facilitating more effective prevention technologies
- Communicating effectively with customers when security issues are identified in released software

You cannot manage what you cannot measure. Unfortunately, implementing an effective metrics monitoring effort can be a difficult undertaking. Despite this, met-

rics are an essential element of your overall application security effort. They are crucial in assessing the current security posture of your organization, help focus attention on the most critical vulnerabilities, and reveal how well—or poorly—your investments in improved security are performing.

- Monitor security metrics
- Gauge the likely security posture of the ongoing development effort
- Enforce accountability for inadequate security

You will see more on metrics and metrics models in the next section on measuring progress.

Security does not end when an application is completed and deployed in a production environment. Making the most out of existing network and operational security investments requires that you inform and educate those charged with monitoring and managing the security of running systems. The following advice and guidance on the security requirements will help to assure that your organization makes the best use of the capabilities you have built into your application.

- Build an operational security guide
- Provide stakeholders with documentation on operational security measures that can better secure the product
- Provide documentation for the use of security functionality within the product
- Specify a database security configuration
- Define a secure default configuration for database resources that are deployed as part of an implementation
- Identify a recommended configuration for database resources for that are deployed by a third party

Development organizations should be bought into the process which they use for development. The most effective way to do that is to build a process engineering team from members of the development team so that they can have ownership in creating the process.

Here are the recommended steps to form the process engineering team:

Build a process engineering mission statement

Document the objectives of the process team. It is reasonable to have the entire development team sign off on the mission, so that those people who are not on the team still experience buy-in and inclusion.

Identify a process owner

The process team should have a clearly identified process "champion," whose foremost job is to set a direction and then evangelize that direction. Make it clear that this team will be held accountable for all aspects of the engineering and deployment activities associated with early adoption of this new security process framework.

Identify additional contributors

As with the process owner, people who make good evangelists should be valued as well as people who will be the most worthy contributors.

Document roles and responsibilities

Clearly document the roles and responsibilities of each member of this team.

Document the CLASP process roadmap

It is time to make the classic "build-versus-buy" decision

for a process framework. Can one of the process roadmaps that are a part CLASP be used as-is? This decision and the resulting process roadmap must be documented and approved before moving into the deployment phase.

Review and approve pre-deployment

Institute a checkpoint before deployment, in which a formal walk-through of the process is conducted. The objective at this point is to solicit early feedback on whether or not the documented framework will indeed meet the process objectives set forth at the beginning of this effort. The team should not proceed to the deployment phase of this project until organizational approval is formally issued.

Document any issues

Issues that come up during the formation of the process engineering team should be carefully documented. These issues will need to be added to the process engineering or process deployment plans – as appropriate to managing risk accordingly.

While any SDLC methodology–with the appropriate security activity in place from start to finish–will do, what should be clear is that metrics and measurement are vital to assure you are headed in the right direction.

Measuring Your Secure Development Program's Success

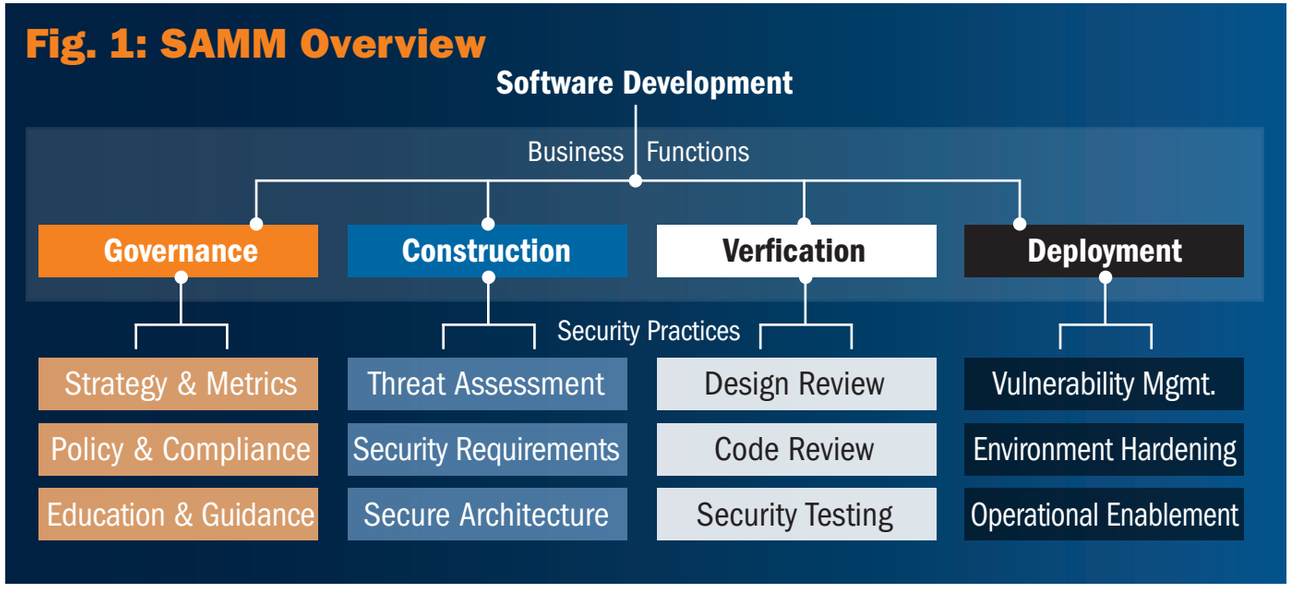
In the last section of this article we'll take a look at the two leading software security maturity measurement models, OWASP's Open Software Assurance Maturity Model (OpenSAMM) and the Building Security in Maturity Model (BSIMM).

OpenSAMM

SAMM is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization. OpenSAMM offers a roadmap and well-defined maturity model for secure software development and deployment, along with useful tools for self-assessment and planning.

SAMM was defined to fit any small, medium, and large organizations using any style of an SDLC. The model can be

Fig. 1: SAMM Overview



applied organization-wide, for a single line-of-business, or even on an individual project. OpenSAMM comes as a 96-page PDF file with detailed descriptions of each core activity and corresponding security processes that you can download for free from the OpenSAMM Web site. The OpenSAMM security practices and framework are shown in Figure 1.

Using OpenSAMM, a company can benefit through:

- Evaluating your organization's existing software security practices
- Building a balanced software security program in well-defined iterations
- Demonstrating concrete improvements to your security assurance program
- Defining and measuring security-related activities within your organization

Building Security in Maturity Model (BSIMM)

The Building Security in Maturity Model is designed to help you understand, measure, and plan a software security initiative. It was created through a process of understanding and analyzing real-world data from nine leading software security initiatives and then validated and adjusted with data from twenty-one additional leading software security initiatives.

Properly used, BSIMM can help you determine where your organization stands with respect to real-world software security initiatives and what steps can be taken to make your approach more effective.

BSIMM lists twelve practices organized into four domains. The domains are:

1. **Governance:** Those practices that help organize, manage, and measure a software security initiative. Staff development is also a central governance practice.
2. **Intelligence:** Practices that result in collections of corporate knowledge used in carrying out software security activities throughout the organization. Collections include both proactive security guidance and organizational threat modeling.
3. **SSDL Touchpoints:** Practices associated with analysis and assurance of particular software development artifacts and processes. All software security methodologies include these practices.
4. **Deployment:** Practices that interface with traditional network security and software maintenance organizations. Software configuration, maintenance, and other environment issues have direct impact on software security.

Fig. 2: The Software Security Framework (SSF)

Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy & Metrics	Attack Models	Architectural Analysis	Penetration Testing
Compliance & Policy	Security Features & Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

The means to the end

It makes no difference what path you take to implementing a secure coding initiative—as long as you continue to strive for improvements, your efforts will be rewarded.

To help you get started with BSIMM, there are free resources on the BSIMM website for collecting information in MS Excel and developing a project implementation plan in MS Project. The spreadsheet will help you study, slice, and dice the activity info within the BSIMM, while the Project file will enable you to copy or click-click-drag the activities to arrange them in the phases or groupings you need. Once you have completed your own assessment, you can build spider diagrams from the results and begin comparing them to others in the same or similar industries. The BSIMM Begin survey tool is also a helpful tool to get started with BSIMM. The survey is a Web-based study focused on 40 of the 110 activities covered in the full BSIMM that lets you walk away with some idea of how your basic software security activities stack up against those practiced by other organizations.

Software Security for Managers: Summary

It makes no difference what path you take to implementing a secure coding initiative—as long as you continue to strive for improvements, your efforts will be rewarded. While any methodology to get there will do, metrics and measurement are vital to assure that you are headed in the right direction for secure systems and software.

Don't let the perfect be the enemy of the good. For software assurance, the time to get moving is now!

The authors' book Secure and Resilient Software Development *is available on Amazon.com.*

Security News

Report: Spam is at a Two-Year High

By Tim Greene • Network World

Spam—particularly that with malicious attachments—is exploding, reaching a two-year high.

Spam—particularly the kind with malicious attachments—is exploding, reaching a two-year high overall, which includes the spike last fall just before the SpamIt operation folded its doors, a security firm says.

In fact, spam traffic is about double what it was then, according to M86 Security Labs, which monitors spam levels across selected domains.

"After multiple recent botnet takedowns, cybercriminal groups remain resilient clearly looking to build their botnets and distribute more fake AV in the process," the company says in its blog. "It seems spammers have returned from a holiday break and are enthusiastically back to work."

This report coincides with a recent report from Internet security company Commtouch, which says a spike in email-attached malware has just ended, but that further waves are expected.

M86 says in its blog that most of the spam is generated by the Cutwail botnet. And it found that much of the malicious spam was couched in phony correspondence from UPS, which concurs with Commtouch's findings that UPS spam was much of what Cutwail and Festi are sending.

It says that other bots are sending other forms of malicious attachments. The Asprox botnet, for example, is sending malicious hotel transaction spam with password-stealing and phony antivirus malware attached, M86 says.

Overall the top subjects of the spam were pharmaceuticals, gambling and dating, M86 says. •

Security News

DHS Releases Software Security Scoring System

By Jaikumar Vijayan • Computerworld

The goal is to give software developers and enterprises more information on finding and fixing the most dangerous programming errors

The Department of Homeland Security (DHS), along with the SANS Institute and Mitre, recently released a scoring system designed to help enterprises verify whether the software they are using meets reasonable standards for secure coding.

The organizations released an updated list of the Top 25 most dangerous programming errors found in software, and a measuring system that lets enterprises score the security of their software based on the presence or absence of those flaws. The goal is to give enterprises information that will let them make more informed decisions regarding the security of their software, said Alan Paller, director of research at SANS.

The hope is that organizations within the private sector and government will use the Top 25 list and scoring system during the software procurement process, he said.

"Companies and not-for-profits that build or buy Web services and software do not have a reliable way to know whether the software they are using is protected against common attacks," Paller said.

The key missing ingredients have been a credible, validated list of the most dangerous errors programmers make, and a way to test the software to see whether those errors are present, he said.

"The DHS/Mitre announcement ... is just that – an updated, authoritative list of the key flaws plus a measuring system that lets organizations score their software for security," Paller said. "The bottom line is that buyers and builders of software and services will be able to ask for assurance that the critical flaws have been eliminated, and be able to verify that."

The updated Top 25 list of most dangerous programming errors that the scoring system is based on includes many of the same security issues from last year's list. The one key difference is that SQL Injection errors top the list for 2011, compared with last year, when they were the second most dangerous error.

Operating System Command injection errors, which allow attackers to issue OS commands through a web application interface, was listed as the second most dangerous soft-

Now with SQL Injection!

One key difference for 2011 is that SQL Injection errors top the most dangerous programming errors list, compared with last year, when they were the second most dangerous error.

ware programming error in this year's list. Rounding out the top five threats were buffer overflow errors, cross site scripting flaws, and missing authentication for critical functions.

The list of errors was accompanied with suggestions and guidance on how software developers can mitigate the chances of such flaws showing up in their products.

"[These] kinds of list are good ways to focus attention on the biggest vulnerability areas," said John Pescatore, an analyst with Gartner. "Things like the Common Vulnerability Scoring Standard have been around for a while providing a common framework for describing vulnerabilities and tailoring severity levels to your own environment."

But what's equally important are ways to measure and drive improvements in the actual implementation of security controls, he said. Efforts such as the Building Security in Maturity Model (BSIMM), for instance, lets companies compare themselves and see how and whether they are improving on the security front, Pescatore said. •

Security Trends

Coming Conundrum: Malware Signed by a Legitimate Developer

By Robert Lemos • InfoWorld

Cyber criminals are stealing code-signing certificates, allowing their malware to get by some defenses

Signed code has become one of the common measures used to secure various computing platforms. Relatively young operating systems—such as Apple's iOS and Google's Android—require that all code be signed using a valid developer signature. More traditional PC operating systems use code signing only for certain system features, such as signing updates and drivers.

Yet cyber criminals and other attackers are starting to use signed code to evade security measures by stealing legitimate certificates from software developers, then using the certificates to sign their malicious programs. Security firm F-Secure posted an example of such an attack, where a malicious PDF file had been signed using a certificate—now expired—issued to a Malaysian government agency.

Signed malware "is problematic, as an unsigned Windows application will produce a warning to the user if he downloads it from the Web—signed applications won't do this," writes Mikko Hypponen, chief research officer for F-Secure. "Also, some security systems might trust signed code more than unsigned code."

In follow-up comments, Hypponen stresses that signed malware is still uncommon and that in 99 percent of cases, a self-signed certificate is used, which typically is not considered trusted.

But data from antivirus firm AVG shows that the nascent problem is growing. In 2009, the company detected about 30,000 malicious programs signed with legitimate—albeit stolen or fraudulently issued—certificates. The next year, that number increased by a third and is on track to triple in 2011.

Developers can help fight the trend by securing their certificate keys, says Yuval Ben-Itzhak, AVG's CTO. "In many software companies, the certificate to sign the code is sitting on the developer's machine in plain text," he says. "Companies should make sure that they are securing the code and the digital certificate from being stolen." •

User Spotlight

Security Manager's Journal: Why Not Spring for Cadillac Security?

By J.F. Rice • Computerworld

Too many corporate decision makers think it's wise to shore up only the most glaring security weaknesses. But attackers will always be able to find the weaknesses you decided you could live with.

Cadillac or Kia? How much security is enough, and how much is too much? Can you even have too much security?

In a performance review several years ago, I was criticized for proposing "Cadillac" solutions to security challenges like patching, security event management, and endpoint security compliance—"Cadillac" being code for "too expensive." It was surreal to hear my striving for excellence put in a negative light. I think what was said in that performance review all those years ago distills a basic conflict between information security and the company it seeks to protect. So, is seeking perfection in security a luxury or a necessity? I continue to be urged to consider it the former, and I continue to see that as folly.

With my current company continuing to suffer increasingly devastating economic pressure in this recession, all IT budgets have been stripped to the bone, we are not going to be able to start any new projects for the foreseeable future, and our main focus is to keep the lights on while spending as little as possible. It is in this climate that I'm struggling to advance my security initiatives, which are necessary to protect the company and save money.

Recently the question of excellence came up again. Our CIO told me that I should start thinking about partial solutions instead of more comprehensive approaches to improving our security. "Instead of trying to solve the whole problem, which is too much for us to handle, just solve a part of it," he told me. I can certainly understand the appeal of that point of view. If we can't afford a full-blown implementation, we can break off a manageably sized piece and focus our resources on that. Makes sense, right?

The problem is, while that reasoning works fairly well within a standard IT environment, it may not make sense in the context of security. If you're talking about converting your old, unstable data storage devices into a state-of-the-art SAN, maybe a compromise or interim solution will tide you over until you can tackle the big project. The same reasoning applies to email (can that Exchange 2010 upgrade wait until

All-or-nothing?

Unlike other IT specializations, where partial solutions can be effective, security has more of an all-or-nothing aspect. Some things we just have to do, or else we risk heavy consequences, up to and including complete failure of the company itself.

next year?) or desktops (upgrade to Windows 7 now, or stick with XP for a while longer?) and many other IT disciplines.

But I'm not convinced the same logic works in the context of security. I've had a lot of time to think about excellence and how it applies to security. Unlike other IT specializations, where partial solutions can be effective, security has a lot more of an all-or-nothing aspect. There are some things we just have to do, or else we risk heavy consequences, up to and including complete failure of the company itself. Security is important to the continuing operation of the company. If we try to save a few bucks by cutting our security budget, we might end up with a breach that could have been prevented, leading to loss of customer confidence, bad publicity, lack of compliance with legal regulations, theft of our confidential data by a competitor, or worse. But those worst-case scenarios aren't very compelling to the company's decision-makers right now. All the focus is on tightening our belts, and uncertain consequences in a murky future are not offsetting that.

So should I take the CIO's advice and get the Kia instead of the Cadillac? Sure, it's better than nothing—but I've come to believe that a successful security program requires excellence. Otherwise, the gaps and holes we don't close will be the ones that ultimately cause our downfall. After all, the bad guys only need to find one weak spot to exploit, while I have to build a consistently solid defense. The job of the attacker is always easier than the job of the defender. Cheaping out on security can cost a lot more than it saves. I think we really do need the Cadillac.

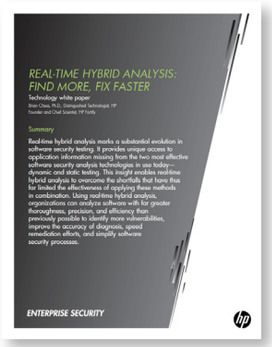
This journal is written by a real security manager, "J.F. Rice," whose name and employer have been disguised for obvious reasons.

Tips and tools

Application Security Resources

Seven Steps to Delivering More Secure Software

Software security is a serious problem, and it is garnering more and more attention. However, the processes that go into making an application more secure are relatively immature. Where do you start? This paper provides seven practical steps organizations can take to begin today. Download >>



Expert Guide to Application Security—Real-time Hybrid Analysis: Find More, Fix Faster

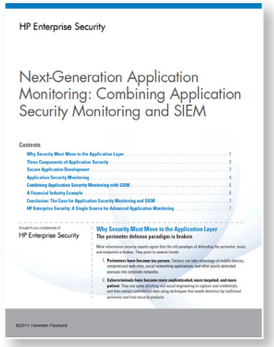
This white paper explores the next generation of hybrid security analysis – what it is, how it works, and the benefits it offers. It also addresses (and dispels) the claims against hybrid, and leaves you with a clear understanding of how the new generation of hybrid will enable organizations to resolve software security issues faster and more cost-effectively than any other available analysis technology. Download >>

Software Security Delivered in the Cloud

The dramatic rise in cyber crime and web application security threats make it more important than ever to know the security state of your essential business applications. This Solution Guide details the automated, turnkey service that requires no special security assessment expertise. Download >>

Does Application Security Pay? Measuring the Business Impact of SSA Solutions

At a time when budgets are under scrutiny, IT is often called on to justify security from a cost-benefit perspective. This Mainstay Partners ROI Study provides the evidence needed for justification of an investment in software security. Download >>



Next-Gen Application Monitoring: Combining App Sec Monitoring and SIEM

Recent statistics show that a majority of security vulnerabilities are caused by security flaws in application and web software. Learn how you can prevent these security vulnerabilities with the “”Next Generation Application Monitoring: Combining Application Security Monitoring and SIEM” whitepaper. Download >>