A CISO’s Guide to Application Security

Learn why application security is more critical than ever to your business, and the six steps to secure applications.
SECURITY BREACHES ARE EXPENSIVE. They cost time and effort in litigation, remediation, and reputation restoration. But when it comes to security, businesses rarely do it right.

Many IT decision-makers tend to focus their security efforts almost exclusively at the network perimeter. They overlook the business applications that run their day-to-day operations—especially partner portals, shopping baskets, or appointment scheduling, which let customers and vendors interact with internal systems. These applications are often packed with credit card and Social Security numbers, addresses, personal health information, or other data ripe for exploitation.

But even focusing on security features at both the infrastructure and application level isn’t enough. Businesses must also consider flaws in their design and implementation. Hackers looking for security flaws within business applications often find them, thereby accessing the hardware, operating systems and data with which the applications interact—without security constraints. In fact, according to Gartner, 75 percent of security breaches are now facilitated by applications. The National Institute of Standards and Technology raises that estimate to 92 percent. And from 2005 to 2007 alone, the U.S. Air Force says application hacks have increased from 2 percent to 33 percent of the total number of attempts to break into its systems.

The longer a business waits to address a security issue, the more costly it becomes. The IEEE Computer Society estimates that a single security vulnerability within an application costs less than $500 to repair in the software design phase. That figure increases to $7,000 if it’s not repaired until the testing phase and $14,000 if it isn’t caught until the maintenance phase. Since each application can contain dozens or even hundreds of se-

<table>
<thead>
<tr>
<th>Cost of Fixing Critical Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost of Fixing Vulnerabilities EARLY</strong></td>
</tr>
<tr>
<td>Stage</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Requirements</td>
</tr>
<tr>
<td>Design</td>
</tr>
<tr>
<td>Coding</td>
</tr>
<tr>
<td>Testing</td>
</tr>
<tr>
<td>Maintenance</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

Identifying the critical bugs earlier in the lifecycle reduced costs by $2.3M

curity flaws, identifying and mitigating them early saves a substantial amount of money—potentially millions of dollars. A recent Forrester Research study says that the average security breach can cost a company between $90 and $305 per lost record (Calculating The Cost Of A Security Breach by Khalid Kark, April 10, 2007). Clearly, application security can’t be overlooked or put off until later.

Some companies understand application security risks and have assigned accountability for minimizing them. The role of Chief Information Security Officer (CISO) has often been established primarily for this purpose. It is also common to have an application security subject matter expert for each development group to ensure that all developers have easy access to solutions to application security problems.

However, practices for identifying security vulnerabilities vary widely. One example is the popular penetration test, or “pen test.” In a pen test, software mimics the actions of hackers from outside the network, and this may be supplemented by “white hat” hackers attempting manually to break application security. Unfortunately, the utility of a pen test is very limited. At the end of the day it can only find problems, not fix them, or even provide assurances that all potential problems have been found. (For this reason application security guru Gary McGraw jokingly refers to pen tests as “badness-ometers”—i.e., they will tell you if your code is bad, but not whether it is good.)

Some businesses approach application security via a web application firewall, which sits on the network and monitors traffic both inside and outside of an application, then analyzes and blocks anomalies in traffic patterns. This technology can be difficult to work with, because the irregular patterns it spots may actually be valid traffic.

To secure your company’s data, your approach must include an examination of the application’s inner workings, and the ability to find the exact lines of code that create security vulnerabilities. It then needs to correct those vulnerabilities at the code level. Finally, it must address the upstream “risk factory” with a comprehensive prevention strategy that fends off future attacks and mitigates current ones.

As a CISO, you understand that application security is important. What steps can you take to avoid a security breach?

6 NECESSARY STEPS TO SECURE APPLICATIONS

1. Find and assess potential vulnerabilities
2. Foster an awareness of risk and the need for remediation
3. Create and deploy application security features
4. Develop continuous methods to find and assess vulnerabilities
5. Secure applications throughout the development lifecycle
6. Make application security an integral part of your operations
STEP 1: FIND AND ASSESS POTENTIAL VULNERABILITIES

✓ **Implement a risk management approach for all applications.** For all but the smallest organizations, the first step in this stage is to deploy an asset management system that creates an up-to-date inventory of every application, including versions, upgrades, patches, and current configurations. This information makes it easier to create correlations with known vulnerabilities and track necessary changes. It also allows for risk ranking of applications and the prioritization of efforts to secure them.

✓ **Identify all software processes that create and interact with a given application.** Examine end-to-end application data flow, identifying the points at which each application interacts with other applications, hardware or data. These are the most likely targets in an attack. Note that application components such as Web 2.0 mashups, service-oriented architectures (SOAs), open source libraries, and even legacy systems are often interspersed within newly developed in-house applications, and may be an integral part of the data flow. Knowing where they are and what is supposed to happen in each component makes it easier to analyze every path that actions and data might follow.

✓ **Prioritize vulnerabilities.** These steps will generate a list—potentially a very long one—of vulnerabilities to address. Don’t start small; the vulnerabilities with the greatest potential impact on business operations and strategy should be top priority. While you can—and should—use them to support a business case for prompt action, it is important to recognize that this initial set of vulnerabilities represents only the immediate problem facing most organizations. Systemic problems arise from the development processes that deliver software to the organization. Often, there are a number of legacy applications in operation that are susceptible to attack.
STEP 2: FOSTER AN AWARENESS OF RISK AND THE NEED FOR REMEDIATION

✔ Alert key stakeholders to the presence of risk and the need to remediate it. It’s not enough to inform key stakeholders that application security is an essential part of overall information security. They need to understand the specific areas where insecure applications threaten the business and the potential consequences—compromised customer information, regulatory violations and the attendant fines, lost sales, damaged reputation—of failing to mitigate them. Once this has happened, CISOs can start to educate the IT staff, especially software developers, about specific security gaps and how to address them. One leading U.S. bank with millions of online customers provided support, resources and incentives to build a code review and application security group. This group, which has veto power over all potentially insecure code applications, reports directly to the CEO—not engineering.

✔ Educate developers. The best way to fix a mistake is to not make it in the first place. Unfortunately, most university programs do not teach developers how to program securely. In one effective educational method, software developers are shown the vulnerabilities in their own work, thus reinforcing the importance of auditing and testing source code at multiple points throughout the development cycle. Fortify has found that an effective developer education program can reduce the introduction of vulnerabilities by nearly 25 percent.

✔ Create an internal marketing campaign. To reinforce the message, both at the IT and corporate levels, consider creating what amounts to an internal marketing campaign with posters, T-shirts, mandatory meetings, and required developer training to “sell” the concept and importance of securing applications. In organizations where security has a strong influence, HR makes it a part of the performance evaluation process.
STEP 3: CREATE AND DEPLOY APPLICATION SECURITY FEATURES

✔ **Develop requirements to secure the application environment.** Begin with clearly defined requirements. Each application, as well as the application environment as a whole, must contain features that follow the security trinity of prevent/detect/correct: the ability to prevent security breaches; detect them when you cannot prevent them; and correct any damages resulting from undetected threats.

✔ **Develop multiple security access control mechanisms for applications.** Today’s applications cannot be secured with passwords alone. New authentication paradigms allow your company to stay one step ahead of the organized criminal element of the hacking community, while Defense-in-Depth is a good way to deal with the growing pervasiveness of spyware.

✔ **Include features that prevent, detect and correct for security breaches in purchased, outsourced, in-house and open source applications.**

1. Purchased applications often contain their own prevention (login) and detection (audit logs) features, but they rarely include remediation and recovery. Ask the developer or vendor about patches for known vulnerabilities, and ask which security patches they have already applied. This gives some indication of how often they need to patch. Before signing any contracts, include a provision that any identified vulnerabilities will be patched immediately. You will need your own infrastructure and operational processes for designing recovery features.
2. Outsourced applications usually include security features, but they may not meet all of your requirements. You must critically analyze the available features and supplement them with internally developed solutions or plug-ins. With outsourced applications that have been customized or custom-built for your company, you must ensure their secure integration with your client identity management system or other access control system.

3. Applications developed in-house must include prevention/detection/correction features built on an approved group of reusable components. All of your internal developers must use the same security components on every application to ensure consistency and minimize the number of lines of code they need to create. These reusable components should be tested multiple times across multiple applications to guarantee across-the-board security.

4. Open source software must follow the same rules as in-house applications, to the extent that your in-house developers can influence the creation and availability of its reusable components. The developers must be part of the open source developer community so they can participate in the process of identifying and patching security issues over multiple releases.

✔ **Patch or code, then test the remediation procedure to ensure it addresses both the vulnerability and the root cause.** Conduct a set of regression tests after adding any patch, checking the entire security feature rather than just whether the patch works. For example, say your login feature includes a bug in which the system treats an asterisk in a password as a wildcard character that lets anyone access the system. With regular testing, developers add a patch, test a handful of passwords with asterisks, see that the system no longer treats them as wildcard characters, and place the patched software into production. With regression testing, developers test every type of username and password to ensure that the system does not now treat some other symbol as a wildcard character.
STEP 4: DEVELOP CONTINUOUS METHODS TO FIND AND ASSESS VULNERABILITIES

✓ Use vendor alerts and public vulnerability databases to track emerging application security issues and assess vulnerabilities on an ongoing basis. Consider creating a job function with responsibility for tracking known issues to prevent anything from slipping through the cracks. The role might also include supervising regular vulnerability assessments. While keeping applications secure requires the same type of review as uncovering initial security flaws, it’s not a one-time process. Your security requirements must be examined for deviations on a regular basis.

✓ Scan and test the security of applications upon each modification to the environment. All new source code should be scanned for known problems, and all old code should be scanned for newly identified vulnerabilities with each deployment. Keep track of past software vulnerabilities and ensure that the mistakes are not repeated in the future. If nothing in your application environment ever changes, keeping abreast of vendor alerts and new security threats should be enough. Of course, since security scans and tests themselves change, you need to monitor emerging issues to determine whether to adopt new testing procedures. Otherwise, test application security at every point for every change, and run a scheduled monthly test to catch unintentional and unauthorized changes.

✓ Provide developers with reliable procedures to scan code for security bugs before compiling, before deployment, and in production. Static analysis products are becoming more commonly used in development. However, it is important to understand that static analysis for quality and security take different approaches. Quality static analyzers are widely recognized as selling to developers who understand the bugs uncovered. Security static analysis, however, often requires what security expert Bruce Schneier calls the “security mindset.”
STEP 5: SECURE APPLICATIONS THROUGHOUT THE DEVELOPMENT LIFECYCLE

✓ Ensure that there is accountability for security architecture and secure application design. A Security Development Lifecycle (SDL) ensures code coming out of the development organization is secure by recommending a set of specific oriented activities be performed during the development process that are tracked and enforced. For example, during the design phase, an SDL will recommend that the security team perform an architecture review to verify that security issues have been addressed.

✓ Provide remediation capabilities that integrate with existing IT tools and processes. Because time is of the essence in security, any application security solution should include tools that facilitate teamwork among security, development and QA staffers, as well as key business stakeholders. Integrating with existing processes, such as the bug tracking system or development environment, improves communication and speeds response time to new security threats.

✓ Incorporate multiple vulnerability detection methods at well-defined control points. An application security solution must work from within the application, capturing data at multiple points for a more integrated view of vulnerabilities. This enables IT staff to look for patterns that indicate which vulnerabilities to prioritize and correct multiple risk points at once. An application security testing solution that encompasses both static and dynamic analysis can detect the broadest range of security holes. Static analysis, which takes place during the coding process, automatically screens an application’s source code and alerts software developers to vulnerabilities. Dynamic analysis operates within an application during testing and production to spot security problems that are only apparent when the code is being executed.

✓ Deploy “toll gates” that block software from going into production until it has been thoroughly tested. “Toll gates” are repeatable, mandatory procedures for finding bugs, which prevent any code from going into production until it, tests free from currently known threats. Your systems development lifecycle should include the application of toll gates to any and all changes to code.
An effective application security strategy addresses both immediate and systemic risk. Most businesses use a wide range of applications—including outsourced, off-the-shelf, open-source, and those developed in-house—each with a different approach to finding and fixing security holes. Moreover, most software developers aren’t security experts, and most security experts don’t develop software. It’s rare for an IT organization to include even one person who is equally versed in both. Bringing application security in-house will most likely require either recruiting employees with an extremely specific skill set, or creating that skill set internally by training existing employees. Both choices are costly and time-consuming.

Software Security Assurance (SSA) is the sum total of the people, process and technologies that can be brought to bear on the problem of application risk in the business. Just as Quality Assurance (QA) is the operational solution to the problem of product quality, SSA is the operational solution to the problem of software risk. It has three distinct elements: measurable reduction of risk in existing applications; prevention of introduction of new risks; and ensuring compliance with application security mandates.

The alternative is to partner with a vendor whose solution has deep expertise in both security and development. This solution should be able to scan the security holes in existing applications and tell developers how to fix them. It should also ensure that code under development is secure at every stage, from design and coding through testing and after implementation, regardless of the type of application. This addresses not only the immediate problem of evaluating and remediating security risks, but also the larger systemic problem—namely, that most software development does not incorporate security into the coding and testing process from the very beginning.
Application Security: The Turning Point?

Are we there yet? CSO Editor-in-Chief Derek Slater wants to say yes, but an OWASP expert says no. By Derek Slater

I wanted very much to write a column about how we’ve reached a turning point regarding application security.

It wasn’t that I thought one particular cataclysmic event has changed our course for the better. Rather, it was an accumulation of smaller observations and developments:

- Writers and bloggers like Jeremiah Grossman, Hugh Thompson, Gary McGraw (and many others) have done great work shedding light on the topic.
- OWASP, the open Web application security project, has established chapters around the world, and its Top Ten Vulnerability list is ever more widely disseminated.
- (ISC)2 recently set forth a new certification covering application lifecycle security issues.
- Both source-code analysis tools and application vulnerability scanners and services can help find flaws on either end of development and deployment. These technologies are maturing quickly.
- And if there is a big one, it would be the application security requirements in version 6.6 of the PCI Data Security Standard, which went into effect this past June and essentially calls for you to use the two approaches mentioned in the preceding paragraph (if not both).

That’s a good bit of app sec activity. Taken together, I thought, maybe it constitutes a quorum of some sort? Alas, as I tried to kindle the flames of a warm and fuzzy analysis of these signs of progress, James McGovern was standing by with a bucket of cold water. McGovern is leader of the Hartford chapter of OWASP. His simple response to my hypothesis: “I think the black hats are winning.”

McGovern gives three reasons. One, companies tend to work toward consensus, which takes time. Even if an application security vulnerability becomes visible to attackers and defenders at the same time, he argues, the attackers are much quicker on the draw while the defenders go through the process of discussion and prioritization. Two, he says outsourced application development creates some obstacles: offshore shops in particular are governed by the rule of margins, so they are discouraged from adding security steps (and therefore time, and therefore cost) to the development process.

Reason three is a bit of a kick in the seat of the pants: McGovern says that technical security is “a hard thing to participate in for nontechnical people,” and that the proliferation of CIOs with nontechnical backgrounds has made it harder to communicate technical risk.

Can’t wait to hear from CIOs on that one. Actually, I’d argue that reason three is really a problem with the communication skills of technical security people; the world isn’t going to grind to a halt so everyone can learn the ins and outs of SQL injection and cross-site request forgery, so the security community is going to have to keep working on nontechnical analogies and other ways of explaining problems.

But at any rate, perhaps McGovern is right, and we haven’t hit an inflection point. Yet. So what’s it going to take? ■

This article was originally published by CSOonline.
Tackling Software Security: An Increasing Threat

Addressing application security solely as an operational issue doesn't work. Attackers are increasingly motivated by financial gain and have been learning how to exploit software for several decades. The same is not true for software engineers, and that needs to change.

By Julia H. Allen

Generally, we think of security as an operational IT issue focused on defending our computers and networks from attackers and security breaches, or we think of information security concerned with protecting sensitive and personal information in digital form. But more and more, the lack of software (or application) security is becoming a greater source of vulnerability for many organizations.

As software and security professionals, we will never be able to get ahead of the game by addressing security solely as an operational issue. Attackers are creative, ingenious and increasingly motivated by financial gain. They have been learning how to exploit software for several decades; the same is not true for software engineers, and we need to change this.

The objective of software security is to build better, defect-free software. Typically software has many defects, and quite a few of these tend to be the source of security vulnerabilities that show up in our operational systems. Software developed with security in mind is more able to resist attack, and in the face of a successful attack, it’s better able to tolerate the attack and recover from it as quickly as possible.

Project managers responsible for software development need to carefully consider the knowledge, skills and competencies of their development team, their organizational culture’s tolerance (and attention span) for change and the degree to which sponsoring executives have bought in (a prerequisite for sustaining any improvement initiative). In some cases, it may be best to start with secure software coding and testing practices. These are the most mature, have a fair level of automated support and can demonstrate some early successes, providing visible benefits to help software security efforts gain support and build momentum. Recommended code and testing practices include:

- Training software developers to implement language-specific secure coding practices and ensuring their use;
- Performing source-code review using static analysis and other types of code-analysis tools;
- Understanding the differences between software security testing and traditional software testing, and reflecting these in the software test program;
- Conducting risk-based security testing that exercises common mistakes, suspected software weaknesses and implemented approaches for mitigating risks to make sure they work and cannot be circumvented.

On the other hand, secure software requirements, engineering, and architecture and design practices offer opportunities to address more substantive root cause issues early in the lifecycle that if left unaddressed will show up in code and test. Recommended requirements engineering and design practices include:

- Using a defined process for identifying and documenting security requirements that includes requirements elicitation, categorization and prioritization;
- Using techniques such as misuse/abuse cases, threat modeling and attack patterns to identify security threats. Attack patterns are a blueprint for creating an attack and include attack prerequisites, related vulnerabilities and the skills and resources required to execute the attack.
- Defining and using assurance cases to capture, communicate, demonstrate and validate desired levels of software security assurance based on defined properties;
Performing an architectural risk analysis to assess the architecture and design’s ability to meet security requirements and resist, tolerate and recover from defined threats.

Regardless of where you start, the following practices are essential when developing software with security in mind:

- Select and integrate security practices (such as those described above) with your existing software development lifecycle and development process—during an acquisition or purchase, requirements specification, architecture, design, implementation, test and deployment. The objective of including security in a defined software development lifecycle is not to overhaul an existing process totally but to add well-defined security practices and security deliverables. Security needs to be tackled in the same way that software engineers address performance and reliability.

- Think like an attacker. In addition to considering what the software should do in terms of functions, features and capabilities, think about what the software should and should not do, and how the software can better resist, tolerate and recover when under attack.

Keep in mind that security is a risk management issue. The highest areas of vulnerability and risk need to be assessed and mitigated during each lifecycle phase. Risks and their priorities will change as the software is designed, developed and deployed. The implementation of these risk management practices depends on the characteristics of the software. For example, risk analysis and assessment for an integrated system have different requirements than assessing risks for a commercial product or an infrastructure component.

Practice selection and tailoring are specific to each organization and each project based on the objectives, constraints and criticality of the software under development.

Julia H. Allen is a senior member of the technical staff within the CERT™ Program at the Software Engineering Institute (SEI), a unit of Carnegie Mellon University in Pittsburgh.

This article was originally published by CIO Magazine.
Recession Increases Security Risks, Particularly Insider Threats

As companies downsize due to economic downturn, they need to keep an eye out for employees disgruntled by layoffs and other cost cuts.

By Jaikumar Vijayan

With a faltering economy resulting in increased job cuts and corporate belt tightening, security analysts are warning companies to be especially vigilant about protecting their data and networks against disgruntled employees.

As it is, one of the biggest threats to corporate data and systems traditionally has come from insiders, who with their privileged access to data and systems, have the potential ability to do more accidental or malicious damage than even the outside attacker.

That threat greatly increases at times when companies are laying off staff, cutting back on raises and bonuses, deferring promotions, consolidating operations and outsourcing work to save money.

One of the biggest threats to corporate data and systems traditionally has come from insiders, who with their privileged access to data and systems, have the potential ability to do more accidental or malicious damage than even the outside attacker.

“All of these increase risk for the company from an insider perspective,” said Shelley Kirkpatrick, director of assessment services at Management Concepts, a Vienna, Va.-based management consultancy.

Tough economic times create uncertainty in the workplace, she said. Employees for instance, can be worried about losing jobs and promotions, concerning about financial liabilities, mortgages and rising energy costs. “When there is uncertainty, it creates stress for employees. It makes the company more vulnerable” to threats, said Kirkpatrick, who was previously a behavioral threat assessment researcher at the Homeland Security Institute.

The threats can manifest themselves in a number of ways. Insiders with access to corporate information, such as customer data or corporate secrets, might want to steal or disclose it for financial gain or simply to get back at their companies. Those with technical-savvy might seek to sabotage corporate data and systems by planting malicious code and so-called logic bombs that are designed to delete data at a future date on critical systems.

The danger is not confined to such actions alone. Stressed, unhappy workers make easy targets for opportunistic rivals as well, Kirkpatrick said. “If I am a competitor looking for a good opportunity to get trade secrets out of my competition, I am going to go after the people who may be stressed emotionally,” she said.

EXAMPLES OF INSIDER SABOTAGE

The damage that insiders with privileged access can do should not be underestimated as several incidents in the past show, analysts said. In July, for instance, a disgruntled administrator for the city of San Francisco locked access to a critical network by resetting administrative passwords to its switches and routers and then refusing to divulge them to officials for days.

In a similar incident, a Unix systems administrator at Medco Health Solutions Inc. who was concerned about being laid off, planted a logic bomb on an internal system that, had it gone off, would have deleted data on 70 servers.

While both incidents involved technically savvy insiders, similar threats can come from non-IT staff as well. In November 2006, a scientist working at DuPont admitted to stealing corporate data valued at around $400 million shortly before he left the company to work at a rival.

The key to being prepared for such threats is knowing what warning signs to look and how to respond to them, said Matt Doherty, a senior vice president at Hillard Heintze LLC, a Chicago-based security consultancy.
One example of a red flag might be an employee who suddenly starts working after hours, stays late for no obvious reason or keeps asking for overtime to make ends meet. Similarly, someone trying to get access to systems and information that they really have no need for could be another sign that something is amiss, he said. Or it could be an employee who prints out large volumes of data after hours, or e-mails it to himself.

As important as such markers are, it is equally important to know what’s going on in terms of employee behavior and morale, Doherty said. Supervisors need to be trained to spot employees in distress or those who could pose a security problem in the future, he said. Companies also need to educate employees about the importance of paying attention to signs of frustration among their co-workers and to have a centralized structure in place for reporting such behavior, he said.

The increased use of portable devices, such as laptops and handhelds, and removable media, such as USB memory sticks and iPods, has also made it easier for rogue insiders to walk away with large amounts of corporate data.

employees about the importance of paying attention to signs of frustration among their co-workers and to have a centralized structure in place for reporting such behavior, he said.

“It’s critical for a supervisor to be aware of the employees, who they are and what’s going on in their lives. It’s really about keeping a finger on the pulse,” he said.

It’s also important to know that the stress can come from outside the work environment, Kirkpatrick said. An employee, for instance, could be experiencing financial problems or may have lost a home to foreclosure because of an inability to meet the mortgage payments.

Identifying and defusing a potential situation takes a coordinated effort, Kirkpatrick said. It’s best for companies to set up a cross-functional team composed of members from the human resources, IT, corporate security, legal and operations departments to deal with potential risks from insiders, Kirkpatrick said. It’s important to ensure that information received about a potential problem is quickly acted upon. But companies need to make sure that any action they take does not violate the employee’s basic rights, she said.

Almost always “there are warning signs. But they are not always listened to,” she said.

Technical controls are vital as well. One of the most important is user authorization and access control, said Raffael Marty, chief security strategist at Splunk Inc., a San Francisco-based company that provides software to help firms search for data in large enterprise applications. Companies that lay off large numbers of people or that engage in a consolidation or merger need to first ensure that former employees no longer have access to internal systems and data, Marty said.

“If a person either leaves his company or is fired, you have to make sure that user account is disabled and that has to happen immediately,” he said. In addition to terminating accounts, it’s also important to monitor critical applications and activity logs to make sure those who previously had access to them can’t access them through some other entry point, Marty said.

It’s a good idea, in general, to monitor privileged user activity to ensure that those with elevated and administrative access rights are not using them to “rob you blind,” added Ted Julian, vice president of marketing at Application Security Inc., a New York vendor of database security tools. “Some sort of monitoring on your most sensitive systems is a must. You need that safety,” in addition to whatever other controls might be in place, he said.

The increased use of portable devices, such as laptops and handhelds, and removable media, such as USB memory sticks and iPods, has also made it easier for rogue insiders to walk away with large amounts of corporate data. Analysts for sometime have said that it’s important for companies to have measures in place for centrally controlling and monitoring which devices can be attached to corporate networks and systems and what data can be downloaded, uploaded and stored on them.

Another category of tools used by companies as a measure against data theft is the so-called data leak prevention tools that keep an eye on network traffic to ensure that protected information doesn’t go outside in an unauthorized manner.

This article was originally published by CIO Magazine.