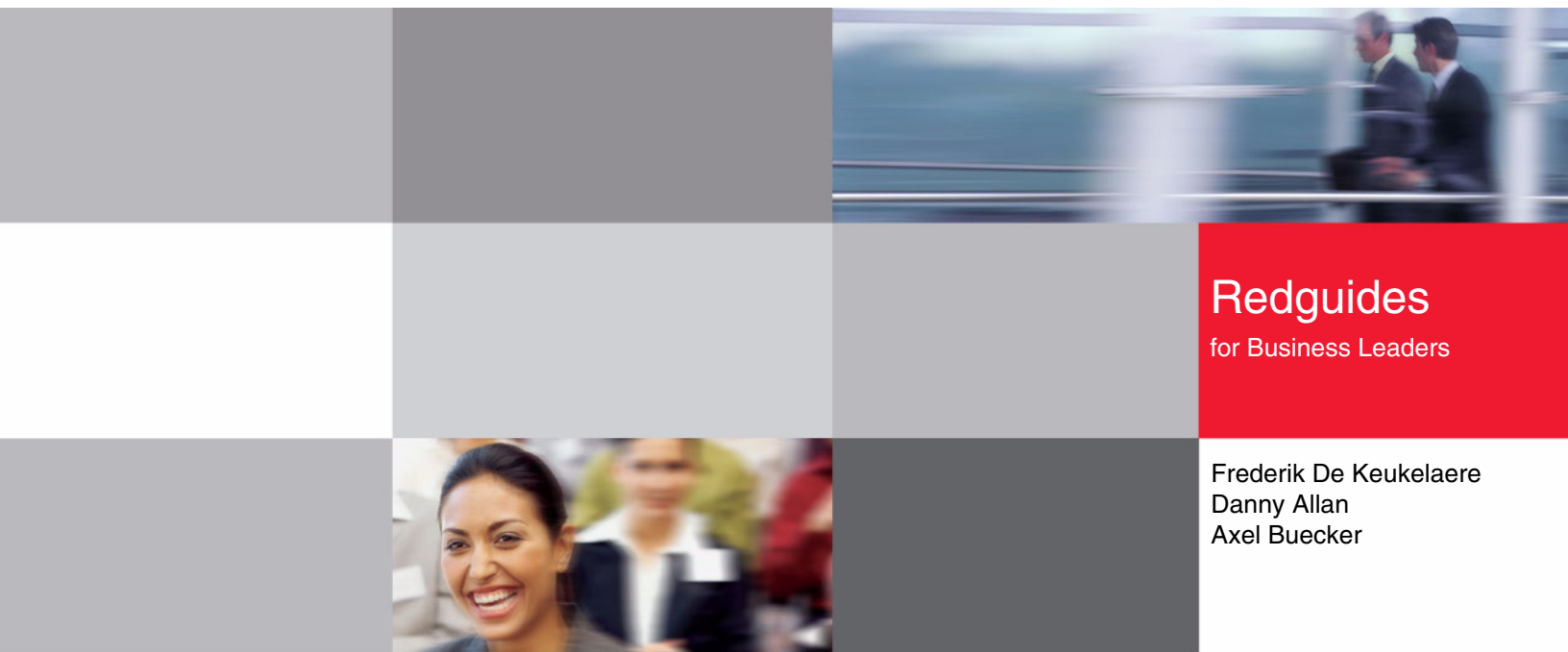


Improving Your Web Application Software Development Life Cycle's Security Posture with IBM Rational AppScan



Redguides

for Business Leaders

Frederik De Keukelaere
Danny Allan
Axel Buecker

- Understand how attackers select targets and turn attacks into money
- Learn the value of automated Web application security testing
- Deploy IBM Rational Web application security tools in your development life cycle



Executive overview

Hackers on the Internet have evolved from fame-hungry sabotage to fraud to profitable organized data and identity theft. As this evolution continues, it is important for business leaders to consider the security of their Web applications as a vital performance indicator of the success of their business.

In this IBM® Redguide™ publication, we explain how your organization can evaluate its risk for hackers entering into your systems. We also explain how your organization can implement security testing and integrate solutions to improve security and protect your information assets.

In the first part of this Redguide publication, we discuss how to evaluate the risk that your organization is exposed to. We explain why your organization is the target of attacks and who is behind them. We illustrate the impact that successful attacks can have on your organization. We show the latest trends and statistics in Web application vulnerabilities and the underground trade of stolen information. We also give a technical overview of the areas where your application can be attacked and discuss the two most common Web application vulnerabilities.

In the next part of this Redguide publication, we introduce the software development life cycle of Web applications and illustrate how security fits into this life cycle. We provide a step-by-step approach to integrating Web application security testing into your software development life cycle. In addition, we also show how and where you can use IBM Rational® products in your software development life cycle to improve the security of your organization based on your business needs.

We conclude this guide with a business scenario in which an organization without any Web application security testing gradually transforms into an organization that delivers high quality secure products.

Uncovering the basics of IT attack patterns and their effect on your organization

The fact that the number of attacks on Web applications is rising is no longer a surprise to most people in the IT business. Successful attacks have received large media attention as have successful arrests. However, what really matters to your organization is understanding the risk that you are going to be exposed to so that you can take appropriate actions to mitigate and control it.

Knowing the answers to the following questions can help you reach that goal:

- ▶ Why might my organization be attacked?
- ▶ How high is the chance that I have a vulnerability in my organization that will be exploited?
- ▶ What will be the damage when my organization is successfully attacked?
- ▶ What can I do to better protect my organization?

Unfortunately, the days of amateurs, college students, or hackers taking joy rides on corporate information systems are largely over. Today's attackers are economically motivated. They are international criminal organizations that make a living by stealing financial information and identities. Today's threat is far more sophisticated and dangerous than the security threats of yore, but in some ways, it is more predictable. Where an amateur hacker might take an interest in any security vulnerability that comes along, serious computer criminals are particularly interested in vulnerabilities that provide a significant return on investment. In short, it is all about the money.

Research about Web application security vulnerabilities published by the IBM Internet Security Systems X-Force® research team shows a tremendous growth of reported Web application vulnerabilities as a percentage of all reported security vulnerabilities over the past years [1]. The probability that these vulnerabilities in your organization will be exploited depends on the complexity of exploiting the vulnerability. Because current attackers are typically for-profit organizations, most of the attacks are done in an automated fashion to keep costs as low as possible. Therefore, if your Web applications can be easily hacked by using automated tools, the chances of exploitation are high.

After your organization has been successfully attacked, the damage attackers can do is vast. Aside from the obvious damages because of data loss, you can be exposed to large fines levied by the credit card companies, have high expenses for notifying cardholders, sustain significant brand damage due to negative publicity, or be involved in civil action against your company. Damages sustained this way quickly go up into hundreds of millions of dollars for large companies [2].

Fortunately improving your Web application security up to a point where you are no longer an economically viable target is not an impossible task. In this Redguide publication, we provide answers to these questions. By integrating Rational AppScan® products into your software development life cycle, we show how you can protect yourself against many of the threats that your organization is currently facing.

Knowing your attackers

Different types of attackers have different motivations for attacking your company. A first group of attackers, called *script kiddies* (also known as *H4ck0rZ*), can target your company if it is a high profile company and attack you, giving them a lot of visibility in the hacker community.

A second group of attackers, called *targeted attackers*, can attack your organization for principles and beliefs, espionage, or political motivations. This group typically has well-defined goals that they want to achieve and choose their targets accordingly.

A third group of attackers, called *organized crime*, can make a business out of attacking any organization with weak defenses and can turn their attacks into money. They do not care specifically about your organization, but if they can, they will use your organization for their own profit.

To learn about a successful U.S. FBI sting operation to shut down a major international, underground Internet forum for buying and selling credit card data used for identity fraud, see the following address:

http://news.cnet.com/8301-1009_3-10234872-83.html?part=rss&subj=news&tag=2547-1009_3-0-20

Script kiddies

The first group of attackers are the hackers that revel in making it into the media. These hackers attack high visibility targets in hopes of making a name for themselves within the hacker community or just hack Web sites for fun. Common types of attacks are defacement and denial of service attacks by which they hope to make a name in the community.

Among the most famous Web application attacks in this category is the MySpace Samy worm in 2005 [3]. The author of this worm used a cross-site scripting (XSS) attack to create a worm that propagated throughout the MySpace social network and in less than 24 hours “Samy” had over 1 million “friends.” It was both a sophisticated attack and by now a well documented one that has become a case study in the Web application security field. Although the Samy worm was merely created for fun, MySpace had to be taken offline to remove the worm.

This example illustrates how this group of attackers can attack your company if you are potentially going to give them the necessary visibility. They go through quite an extent to have their own practical joke while creating publicity for themselves along the way.

Targeted attackers

A second group of attackers is operating in quite a targeted way. Typical examples are attacks for espionage (nation, state, and corporate), political or religious beliefs, and so on. Targeted attackers are often hired by organizations that use the Internet as a front for conducting their wars. These organizations typically steal specific data and intellectual property, or try to spread their political or religious beliefs.

The security community has been monitoring several propaganda wars. For example, in his blog, Gary Warner [4] writes about several of these propaganda cyber wars:

The original cyber propaganda war was launched by Chinese hackers in May of 2001 after the collision of a Chinese fighter jet with a U.S. Navy plane. Tens of thousands of U.S. Web sites were defaced by Chinese hackers blaming the U.S. for the incident. More recently the technique has been adopted by Muslim hackers, beginning with the defacement of thousands of Danish and American Web sites in February 2006 after the publication of cartoons about the prophet Muhammad, and against Israeli and U.S. Web sites after the bombardment of Lebanon by Israel in August of 2006.

The targeted attackers are a real threat to your organization if it is a government institution or in any way the focus of the organizations behind these attackers. They put a lot of effort in attacking the specific targets that help them achieve their goals.

Organized crime

The third group of attackers is called organized crime. The often made assumption that “if there is no specific reason why attackers would be interested in my company, they will not attack me” does not hold for this group. Computer criminals that are part of organized crime look for information that they can quickly turn into profit. By and large, this quick turn on investment means consumer credit card information and bank account access credentials.

While sometimes attackers find ways to harvest vast amounts of this type of data from corporate servers and networks, a great deal of this information is stolen by spyware running directly on user PCs. Corporations with their advanced patching and protection mechanisms create obstacles for these attackers. However, consumers, with their lack of protection, casual patching behaviors, and general lack of security, remain easy targets. Organized crime uses an organization’s servers as a launch platform for their attacks to target customers. For example, they might use your forums to spread malicious files, such as specially crafted PDF files and multimedia applications (for example Flash) that contain embedded exploits for installing the malware on your customers’ PCs.

In addition, certain types of corporate applications, namely custom-built software, such as Web applications, remain highly-profitable and inexpensive targets for criminal attackers. Consider the sheer number of new vulnerabilities that are found in commercial and open source Web applications, the majority of which have no available patch. When coupled with the hundreds of thousands of custom Web applications that are also vulnerable (but never subject to a vulnerability disclosure, much less a patch), they become the Achilles heel of corporate security. Attackers continue to target Web application vulnerabilities, especially Structured Query Language (SQL) injection, to plant malware on unsuspecting users that visit vulnerable Web sites.

It is safe to assume that organized crime that is trying to make money from attacking computer systems does not care about your systems specifically. However, if your applications are easily exploited by using automated tools, you are high on their target list.

Criminal economics 101

To gain a better understanding of how criminal organizations make money by attacking an organization, this section looks at criminal economics.¹ On a basic microeconomics level, an understanding of the opportunity for a computer criminal comes from considering the amount of revenue that can be generated from exploiting a vulnerability relative to the cost of doing so. Obviously, vulnerabilities that present a high revenue opportunity at a low cost are likely to be popular with attackers. Both revenue (opportunity) and cost are made up of a complicated set of components. Some of these components can be influenced by the security of your applications.

Criminal opportunities

The actual revenue that can be generated from exploiting a vulnerability is a combination of the size of the installed base of vulnerable hosts and the value to an attacker of controlling each host. This is usually due to the information that the hosts contain and the price that the attacker can ask for that information on the black market.

When a vulnerability is first disclosed, the installed base of vulnerable hosts can be quite large. If the value of controlling those hosts is also large, the attacker has a significant theoretical revenue opportunity. This sort of situation can motivate efforts by the security industry to roll out patches quickly and reduce the size of the installed base. If the industry is

¹ The Criminal economics discussion is in large an excerpt from the IBM Internet Security Systems, “X-Force 2008 Trend & Risk Report” [1].

effective, the total real revenue opportunity can quickly become too small for attacks to materialize. However, there might be cases where the installed base of vulnerable hosts is large, but the value of controlling the type of host that typically runs the vulnerable software is small. Therefore, attackers have little incentive to exploit the vulnerability regardless of what the security industry does to remediate it.

Criminal costs

The cost of generating revenue by exploiting vulnerabilities is also made up of several factors. First is the cost of obtaining an exploit, which depends on whether an exploit is publicly available. Second is the difficulty associated with using the exploit. As is the case with legitimate businesses, criminal organizations have operational processes that are built up around repeatable sets of circumstances and automatable tasks.

Vulnerabilities that fit into existing processes and that can use existing automation are easy for criminals to monetize. Vulnerabilities that require the development of new processes or software are much less likely to present an attractive opportunity to criminals, particularly if they represent a one-of-a-kind set of circumstances that is unlikely to be repeated in the future. Even when it makes sense for criminals to develop a new attack methodology to exploit a new class of vulnerabilities, widespread attacks usually take longer to emerge than vulnerabilities that fit directly into an existing process.

Making money

Just as in any business, criminal organizations must calculate the value of a potential attack to judge whether it is worthwhile to pursue. To compare calculating the value of an attack with calculating the value of creating a Web application, we first look at the value of a Web application. How do we determine the value of our Web applications?

Value of presenting information to potential customers?	\$10
Value of reducing costs through account self-service?	\$100
Value of delivering business functionality in a rich UI?	\$1000
Value of creating a rich, collaborative, user community?	Priceless!

From this calculation, it becomes clear that the rich, collaborative, user community is the way to go if criminal organizations want to make money in this business. Pursuing the rich, collaborative user community will most likely happen despite the security concerns the fearful security person might have. This person will try to stand in the way of rich interactive environments because they can pose a real threat. However, trade-offs must be made to correctly recognize the value.

When looking at the business model of a criminal organization, we can do a similar calculation to discover its value:

Value of stealing a single credit card number?	\$0.10
Value of stealing a single e-mail password?	\$4
Value of stealing a person's bank account credentials?	\$10
Value of an automated random attack system collecting tradable information?	Priceless!

Since criminal organizations often make money from trading the information they get by attacking Web applications, we use the current trading values of these types of information as shown in Table 1 on page 6 to make our calculation.² The table shows information about different goods, how often they are requested and sold, and their price range.

² The information in Table 1 on page 6 was published by Symantec™ in *Symantec Report on the Underground Economy*: November, 2008, p. 20 (http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf). Reprinted by Permission.

Table 1 Goods and services for sale and requested

Rank for sale	Rank requested	Goods and services	Percentage for sale	Percentage requested	Range of prices
1	1	Bank account credentials	18%	14%	\$10 - \$1.000
2	2	Credit cards with CVV2 numbers	16%	13%	\$0.50 - \$12
3	5	Credit cards	13%	8%	\$0.10 - \$25
4	6	E-mail addresses	6%	7%	\$0.30/MB - \$40/MB
5	14	E-mail passwords	6%	2%	\$4 - \$30
6	3	Full identities	5%	9%	\$0.90 - \$25
7	4	Cash-out services	5%	8%	8% - 50% of total value
8	12	Proxies	4%	3%	\$0.30 - \$20
9	8	Scams	3%	6%	\$2.50 - \$100/ week for hosting \$5 - \$20 for design
10	7	Mailers	3%	6%	\$1 - \$25

Since the trading prices of individual pieces of information is relatively low, the real money is made by trading large volumes of data. Therefore, attackers must make a trade-off between the cost of attacking and collecting data and the value of that data on the market.

Just as in any business, automation is a tool that is used for cutting the costs. For that reason, organized crime poses the biggest threat to any organization since organized crime does not care who it steals from, as long as it can keep costs low.

This brings us back to the probability of the exploitation of an existing vulnerability in your Web application. If you can place a check mark in each of criteria shown in Figure 1 for your organization, consider the chances that an automated tool hacks into your application to steal information or use you as a malware distributor significantly high.

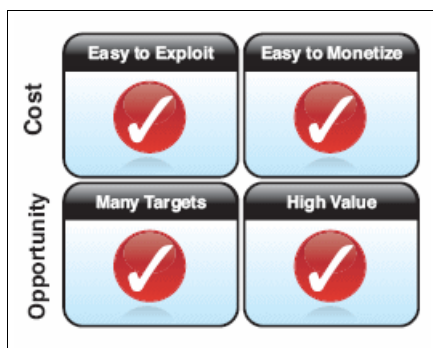


Figure 1 Exploitation probability of a vulnerability in your Web applications³

³ Published by IBM X-Force in January 2009.

Impact on your organization

Now that we have a clear understanding of the motivations of a criminal organization to target any organization, including yours, we look at the kind of damage that successful attacks can do to your organization. Unfortunately, there are many ways in which your company can sustain damage from successful attacks. Data loss, brand damage, and unknowingly helping criminals are three of the most common ways in which your organization can be impacted by successful attacks.

Data loss

A directly noticeable impact of successful attack is data loss. Data loss occurs when attackers break into your system and steal data from your machines after which they delete. When you use your machines after the attack, this data is no longer available to your business, which can make it impossible for you to continue doing your current business the way you have done it before. Imagine the impact of losing all your customer records and how this might be a tremendous loss for your company.

Brand damage

A bit more long-term damage is the damage that attackers can do to your brand. Attackers can potentially deface your Web site, steal a huge amount of your data, make it publicly available, and so on. All of these actions can potentially receive widespread attention and reach your customers. This can give your brand a negative image and make it harder for you to keep existing customers or to attract new ones.

Unknowingly helping criminals

Even if criminals do not steal any data or deface your Web site, they can still use your machines as hosts for spreading their malware. If this is discovered and made publicly available, this can lead from brand damage as explained in the previous section. However, even if this is not made publicly available, you are unknowingly helping criminals to attack your customers, which can have tremendous legal repercussions on your business.

Trends and statistics

Unfortunately quantifiable numbers regarding security breaches are difficult to find because many companies prefer to keep this kind of information secret. The main motivation for secrecy is the brand damage they might incur from revealing the specifics of the breaches. However, some alarming security trends and statistics have been reported in public forms, which we discuss in the sections that follow.

Increasing focus on Web applications

Research published by the IBM Internet Security Systems X-Force research team [1] shows the tremendous growth of reported Web application vulnerabilities as a percentage of all reported security vulnerabilities over the past five years. In fact, more than 54% of all vulnerabilities found since 2006 can be in commercial and open source Web applications. This percentage should not be surprising given the enormous push to more broadly enable business functionality through the Web and drive growth in this challenging market.

As a result, several Web application vendors have made it to the top 10 list of the vendors with the most vulnerability disclosures in 2008, where this list previously was dominated by larger

non-Web related vendors. Figure 2 illustrates the explosive rise in discovered Web application vulnerabilities between 1998 and 2008.

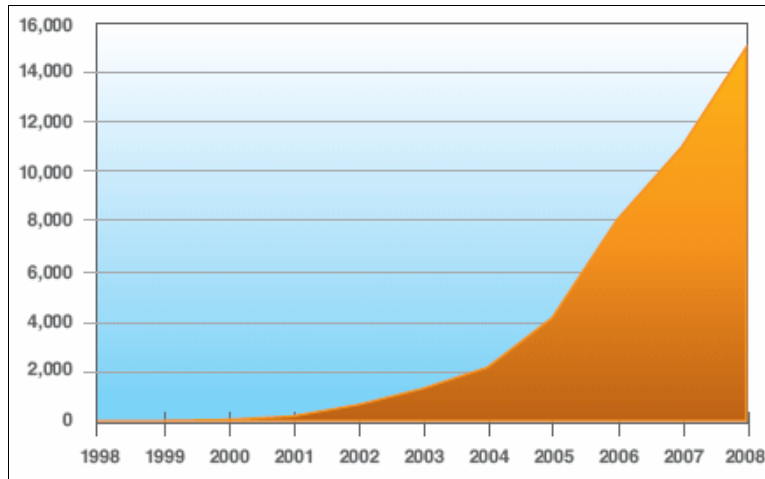


Figure 2 Cumulative count of Web application vulnerabilities⁴

According to Mitre Common Vulnerabilities Enumeration (CVE) [5], a list of known security vulnerabilities sponsored by the National Cyber Security Division and the U.S. Department of Homeland Security, the total number of publicly reported Web application vulnerabilities has risen sharply. The number has risen to the point where it has overtaken buffer overflows, which is one of the most commonly found security vulnerabilities in software. This increase is most likely because of the ease of detection and exploitation of Web vulnerabilities, combined with the proliferation of low-grade software applications written by inexperienced developers. In turn, the increase is caused by the ease of creating small simple Web applications. Despite this simplicity, developing secure high-grade Web applications remains a complex problem.

In Web application security, there are a couple of often reoccurring vulnerabilities. The top three most important vulnerabilities are SQL injection, XSS, and file include. Figure 3 shows an overview of the trends of these vulnerabilities between 2004 and 2008. You can see how the top three vulnerabilities take up to 70 to 80% of all discovered vulnerabilities.

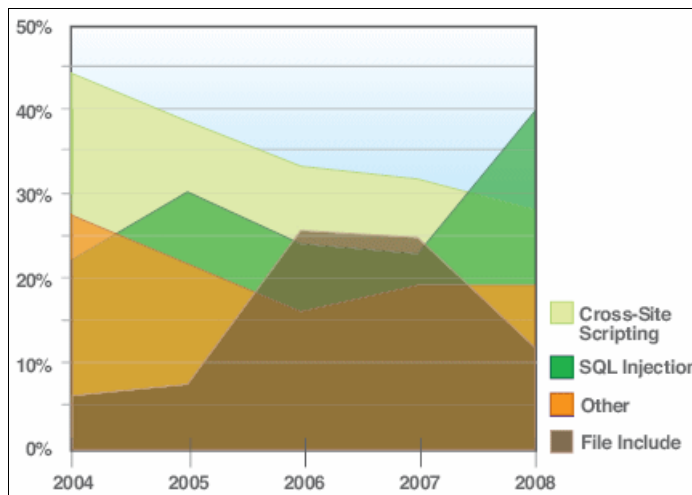


Figure 3 Web application vulnerabilities by attack technique, 2004 - 2008⁵

⁴ Published by IBM X-Force in January 2009.

⁵ Ibid.

In this guide, we restrict the discussion to SQL injection and XSS because they were respectively the number one and number two vulnerabilities in 2008. For further information about file include, see the *IBM Internet Security Systems X-Force 2008 Trend & Risk Report* [1].

Probability of attack

The question remains: What percentage of these vulnerabilities will actually be exploited in an attack? The hard reality is that malicious attackers have also noticed the omnipresence of Web application vulnerabilities on security issues lists. Gartner has estimated that 75% of online attacks are now focused on Web applications [6]. Couple this statistic with the alarming realization that most security dollars are spent on the underlying network, not on the application, and there is evidence of a potential problem. Web applications have become a target of attack, regardless of whether it is for financial gain, political motivation, or a means for some individuals to have fun.

Likelihood of vulnerability in custom Web applications

Based on the previous statistics, it has become clear that security issues are often found and reported in commercially packaged or open source software. However, this does little to help the organization understand the likelihood of vulnerabilities in outsourced or internally developed applications. A broad analysis of 32,717 applications in 2007 by the Web Application Security Consortium (WASC) [7] revealed that 96.85% of these applications had at least one high severity vulnerability. XSS vulnerabilities were found in 58.11% of the applications, and SQL injection vulnerabilities were found in 25.30% of the applications. Therefore, the chances that there are vulnerabilities in your custom Web applications should be considered significantly high.

Pinpointing Web application weaknesses

In this section, we provide technical background on the weak points of Web applications. We give an overview of the most significant places where Web applications can be attacked and provide insight into the details of two of the most common attacks on Web applications.

What can go wrong in Web applications?

Web applications have several areas in which things can go wrong. The main trouble points are unsafe data transfer, lack of input validation at the server-side, and lack of control over the client side.

Unsafe data transfer

Data can be stolen as it crosses the network. In most Web applications, the data packets that pass between the client and server are transmitted across an uncontrolled and untrusted network. For example, if you open a browser and request a page from a stock trader, the request is broken down into packets and transmitted across the public Internet, with numerous routers and switches in the middle. While scalable, the model used for Internet transport leaves the data vulnerable to man-in-the-middle (MITM) [8] attacks. In this type of attack, the attacker is in between the starting point and endpoint of the data transmission and monitors or modifies the data it passes. For example, an attacker can modify the content of the stocks news, hoping to influence any decisions the receiver might take based on the falsified content.

A non-technological parallel is your mail carrier. Even if you know and trust the individual who picks up your mail each day, you do not know what happens between the time your mail leaves your mailbox and the time it arrives at its destination. For example, the letter might

have been steamed open or exposed to light imaging technology. Also, another person might be involved in the transit of that one letter. In both cases, security technologies are necessary to protect the integrity of the content during the transmission.

Lack of input filtering at the server side

Server-side application security is a large and complex topic that cannot be exhausted in this guide. An application, by definition, receives input from the user and takes action based on that input. The input might be a page request, a key stroke, a click, or a form submission. Each of these inputs is processed by the server. If the input is not handled securely, it might be used to subvert or manipulate the application in unintended ways.

We can extend the example of the mail carrier to include an attack on the mail infrastructure. Suppose that the mail carrier does not properly filter incoming mail packages by properly x-raying it and checking it for explosives. This mail carrier can be attacked by sending a package with a bomb inside. It is easy to imagine how improper filtering of mail packages can have a devastating impact on the mail carrier.

Lack of control over the client side

Although the server side used to be a popular place of attacks, an increasing number of attacks are no longer directed at the server, but at the client. This is especially true in newer Web 2.0 applications because these applications have moved more of the logic toward the client side to increase the responsiveness of the applications. While you might have strong security controls over your application code and environment, it is almost impossible to control the behavior of either the client individual or the client machine. As a result, many things can go wrong.

Again we further explore the mail carrier example in light of client-side attacks. Suppose that the attacker has a specific receiver in mind. This receiver might be attacked by creating a specially crafted letter, for example one that contains anthrax, that will attack the receiver as soon as the letter reaches its destination.

Web application vulnerabilities

In the following sections, we look at two of the most used Web application vulnerabilities and point you to where you can find a fuller list of vulnerabilities.

SQL injection

Since a variety of documents exist on the Web that explain SQL injection extensively [9-11], we do not go into details about all aspects of SQL injection. However, by means of an example, we illustrate how SQL injection works and show how it can impact your organization.

Consider an application with a custom login form that includes a user name and password. In order for the system to authenticate the user, a query is sent to a database that says: “Does user *X* have password *Y*?” If the result of this query is *true*, the user *X* is authenticated. If the result is *false*, the user is rejected.

Now suppose that the server side did not do proper validation and allows somebody to create the following query: “Does user *X* have password *Y* or did I enter *X* as the user name?”⁶ Instead of simply providing *Y* as the password, the malicious user makes sure that the query always returns *true* when considered as one query. Example 1 on page 11 shows the SQL query that is associated with this example.

⁶ To make it visually clear what is happening, we placed the input from the user in italics.

Example 1 SQL query for logging into a Web application

```
SELECT userid, full_name
  FROM members
 WHERE username = 'X' AND password = 'Y'
```

To successfully attack this query, the attacker must ensure that the input provided breaks out of the currently predefined set of commands in the SQL query (SELECT, FROM, WHERE, AND) and inserts the attacker's own commands. As soon as the attacker is capable of breaking the predefined structure, this person can manipulate the query into returning the desired results. The attacker will have the same permissions as the process for executing the SQL command and can then perform a variety of actions with your database.

Now let us assume that our server does not do any validation and that all input for the user name and the password fields is directly redirected into X and Y. The attacker can then ensure that this query is always true by using single quotation marks to break out of the predefined commands and inject the attacker's own commands.

Injecting malicious input into Y can result in a query similar to the one in Example 2 that can always authenticate the attacker.

Example 2 Malicious input breaks our SQL query

```
SELECT userid, full_name
  FROM members
 WHERE username = 'X' AND password = 'Y' OR 'X' = 'X'
```

The list of things that an attacker can do by injecting malicious input into SQL queries is virtually unlimited. Basically an attacker has the same access rights as the process that executes the query and can also do anything that this process can do. To attack your organization, the attacker creates a variety of input values in attempts to discover which characters you allow in your queries. Afterwards the attacker tries to learn the database structure and attempts to exploit this knowledge to obtain table names, user names, and passwords. The attacker might even destroy the entire database. If your company does not possess any valuable data for the attacker, this person can use SQL injection attacks to inject malware into your pages and use your Web site as a distribution platform for attacks on your Web site visitors. Because of the power and ease of automation, this attack was the most popular type of attack in 2008.

Unfortunately injection flaws are not limited to databases, but can include file system injection, Mail (MX) injection, Extensible Markup Language (XML) injection, Lightweight Directory Access Protocol (LDAP) injection, and server-side include (SSI) injection to name a few. If the server does not correctly handle all application input, the application environment can be compromised, business logic subverted, or protected data exposed.

Cross-site scripting

Because XSS attacks are widespread, a lot of material is available on the Web [12, 13] for this type of attack. We take the same approach as for SQL injection and provide an example that explains how attackers can potentially exploit this type of vulnerability.

Consider a support forum. If the input to this forum is not properly filtered, a malicious user can post arbitrary HTML on the forum. When another user looks at the forum, the HTML code is displayed to the user. If the HTML code contains a script, this script is executed each time a user visits the post on the forum. Example 3 on page 12 illustrates how an attacker can insert a pop-up message that is displayed to anyone who visits this post.

Example 3 Message posted unfiltered on a forum in which an alert is displayed for every viewer

```
Hello everybody,  
Thank you very much for helping me out. Due to your suggestions I was able to  
figure out how to solve my problems.  
Thanks, Evil Bob  
<script> alert("Don't you like Evil Bob?");</script>
```

At first sight, it does not appear that XSS vulnerabilities in your Web site can have a high impact on your users. However, if XSS attacks are taken to their maximal extent, they can be just as dangerous as SQL injections or maybe even more dangerous. Depending on the sophistication of your attacker, there are different levels of control over your users that the attacker can acquire by injecting code into your pages:

1. The most basic attacks inject pieces of data into your Web pages in order to deface them. Typical examples of this are injecting messages such as “you’ve been hacked,” inserting pictures, and so on.
2. At the next level, your attacker might try to influence your user by providing wrong or misleading information. An example is providing false information of the evolution of your stocks, falsifying your online annual reports, and so on.
3. At a third level, the attacker might try to interfere with the normal usage of your Web site. The attacker might use the XSS vulnerability to inject code that makes the Web site behave in a strange way or makes it unusable. Examples of this are generating thousands of pop-up messages, redefining where your links point to, and so on.
4. A more sophisticated attacker might exploit XSS to steal identities. By injecting code that runs on the page while the user is logged in, the attacker can use a key logger to intercept passwords and steal your users’ identities.
5. At the fifth level, the attacker monitors your users as they use your Web site. An advanced XSS attack allows an attacker to monitor every action your user is taking. The attacker sees every page that the user sees and possess all the information that the user has entered (including their user name, password, credit card information, addresses, and so on).
6. In the final step, the attacker can fully take control over the user’s environment. By controlling the user, the attacker can make decisions on his behalf, navigate to different parts of your Web site, purchase goods, and so on.

Other vulnerabilities

The full list of vulnerabilities to which Web applications are currently exposed is much longer than we can describe in this guide. For more information, see the Web pages of the Open Web Application Security Project (OWASP) [14] and the Web Application Security Consortium (WASC) [15].

Protecting your Web applications from attacks

Similar to the way in which we must look at every aspect of the Web application to find the highest number of vulnerabilities, is the way that we must cover every aspect of the software development life cycle to build our Web applications as secure as possible.

In the first part of this section, we give an overview of the different pain points in the software development life cycle, introduce the different ways to test software security, and highlight the two most important best practices. In the second part of this section, we start with a general introduction of the IBM Rational tools that can assist you in securing your software

development life cycle. Then we go into detail about the IBM Rational AppScan product line, describe their different versions, and explain how to integrate them into your software development life cycle. In doing so, we show you how you can build a security ecosystem for developing your Web applications.

Securing the software development life cycle

The software development life cycle consists of three phases:

- ▶ Design phase
- ▶ Development phase
- ▶ Delivery phase

Each phase plays a role in the quality of the overall security of your final product and, therefore, must be considered from a security perspective.

Design phase

The design phase in the software development life cycle consists of creating requirements and designing the architecture of the application. To secure the software development life cycle, both the requirements and architecture design must be performed with security in mind. Almost every application suffers the potential of greatest weakness if the requirements and the architecture are not clearly designed, planned, and executed.

Within the scuba diving community, there is an important mantra, “plan the dive, and dive the plan.” Failure to plan can result in serious consequences. While human life is not usually at stake, the same principle holds true for Web applications. The most catastrophic failures in software have occurred when the plan for the software is not robust and secure by design. Such a plan can be created by having a decent set of requirements and a design that fulfills them. To assist with this process, various IBM Rational tools can be used.

Consider a common scenario that relates to authentication. Research in the United States has shown that one in nine people uses one of the 500 most common passwords and that one of every 50 people uses one of the top 20 passwords. This is a big security problem, because for hackers, it is easy for them to use brute force passwords if they are on the 500 most common passwords list.

The commonly used way to counter this problem is to lock out any account that has too many failed attempts in a short period of time. However, hackers can try to avoid lockout by trying the most common passwords with different user names. There is little you can do since you do not want to lock out all accounts. Nor do you want to disable access from the attacking IP address out of fear that this will block legitimate users from coming through the same gateway.

Therefore, designing an application securely is essential. Having a requirement that your system cannot accept easy-to-guess passwords might be sufficient to prevent this. Of course, this requirement must be implemented correctly, which brings us to the next phase in the software development life cycle, which is the development phase.

Development phase

The development phase is a three-step process in which code is written, built, and tested. Although many software development groups recognize the need for developing an application securely, experience has proven that developing a secure application is more than a little difficult. In fact, most of the reported vulnerabilities are the result of poor development practices. A typical example of a bad development practice is an inexperienced developer who writes a custom component and along the way introduces a security vulnerability into the application. A much better practice is to use an existing, proven component from a mature framework that has been thoroughly tested for security vulnerabilities. In addition, educating the developer on secure development practices pays off in the future.

With the fast changing rich Web 2.0 UI designs, the imperative for secure code development becomes even more critical. The continuously changing Web 2.0 designs leave little space for thorough testing. On top of this, to maximize interactivity, a large part of the application code is run on the client browser and, therefore, can easily be viewed by the user. The organization must assume that the user will intentionally tamper with the exposed application business logic and try to exploit it to its own advantage.⁷ By integrating the right tools into the development process, many of the security-related tasks can be automated during the coding, building, and testing of your Web applications.

Delivery phase

The most securely designed and developed software fails to be secure when it is delivered in an insecure environment. This includes (but is not limited to) the hardening of the application infrastructure, protection of the data as it crosses the network, the defense of the production environment, and a patching and update strategy for the supporting operating system and components.

For example, failure to configure the Web server to deny access to the directory structure can allow a malicious user to gain direct access to sensitive information and application code. Therefore, a secure delivery phase exists from the final audit of the security of the application in its delivery environment and afterwards maintaining the security level of the operating environment. Again, a wide variety of tools can be used to automate these tasks.

Shades of analysis

The tools used in the software development life cycle for analyzing the security of your Web applications can be roughly divided into three different types of tools:

- ▶ White box analysis tools
- ▶ Black box analysis tools
- ▶ Gray box analysis tools

Their distinction is made based on the amount of information about the system and software that is available to these tools when performing their security analysis. White box analysis has the highest access to this information and can be seen as approaching security from the developer's perspective. Black box analysis starts from the perspective of an attacker with no access to any of this information. Gray box analysis combines both black box and white box analysis to improve accuracy and coverage.

⁷ A good example of hackers reverse engineering business logic based on visible client-side logic is the 2007 attack that allowed hackers to obtain the discount codes for free VIP passes to Macworld 2007 [16].

White box analysis

With white box analysis, all relevant information about the system or software is known and available to the tester. This approach to analysis is common in the quality assurance space where the people responsible for the software have access to both the design documentation and the source code. With the exception of open source software, this information is not generally exposed beyond a defined group of individuals.

Given the relevant information, white box analysis tends to be comprehensive. It can quickly uncover potentially obscured interfaces that are not visible to the outsider. White box analysis can quickly determine the complete attack surface and create the necessary set of tests.

It is important in white box analysis not to put too much emphasis on either the design specification or the source code. Placing too much emphasis on the design specification can lead the tester to miss functionality that has been built outside of this document or that has been implemented incorrectly. Placing too much emphasis on the source code can lead the tester to miss critical exposures that are related to the architecture or the way in which the system was designed.

White box analysis includes the following the three primary techniques:

- ▶ *Architectural analysis*, which is often referred to as *threat modeling*, seeks to enumerate the goals an attacker has within the software and introduces countermeasures for each of these threats
- ▶ *Source code analysis*, which scans the source code and traces user input through the application to find vulnerabilities and bad coding practices
- ▶ *Static binary analysis*, which operates similar to source code analysis but at the binary level, which allows it to find contextual risks and platform-specific problems

White box analysis offers the following benefits:

- ▶ All relevant information is available to the tester.
- ▶ Logical design flaws can be determined.
- ▶ It is simple to quickly document the entire attack surface.
- ▶ It is effective at finding programming and implementation errors.

White box analysis includes the following challenges:

- ▶ Poor coding practices can falsely be detected as vulnerabilities.
- ▶ Software cannot be easily tested remotely in a distributed environment.
- ▶ Access to the design specifications and source code is not always possible.

An example of white box analysis is running a source code scanner during development.

Black box analysis

Black box analysis involves examining the software or system with no prior knowledge of the environment. This type of analysis is similar to what an outside attacker might do. Using automated tools and manual techniques, the analysis begins by detecting the attack surface and probing the system for relevant information.

A fundamental concept of black box analysis is to understand the system as fully as possible before actually testing for security issues. Information disclosure and improperly configured systems can be especially helpful in building this knowledge. This foundational understanding of the software or system allows the tester to be more efficient during the testing phases.

When an organization is sensitive to the outsider threat, black box analysis is often the first analysis approach. The risk findings from this analysis and testing are often prioritized because it more accurately reflects the immediate risk posed by the outsider.

Black box analysis includes the following two primary techniques:

- ▶ *Vulnerability scanning*, which is realized by using a large database of known vulnerabilities and trying to identify a known vulnerability in an application. This can be either passive, for example by searching for a version number in a Web page, or active, for example by trying to exploit a vulnerability and searching for a known exploitation result.
- ▶ *Dynamic analysis*, which is often referred to as *Web application scanning*. It tries to automatically scan and document the attack surface, test the application by means of fault injection, and determine the presence of a vulnerability based on the response. The main difference between this type of analysis and vulnerability scanning is that dynamic analysis can discover unknown vulnerabilities, where vulnerability scanning only tests for known ones.

Black box analysis includes the following benefits:

- ▶ Testing deployed software allows for high levels of confidence in the results.
- ▶ There is no need for access to design specifications or source code.
- ▶ Software can easily be tested across the network.
- ▶ Testing in the deployed environment allows for environmental analysis.

Black box analysis include the following challenges:

- ▶ It is impossible to determine code coverage; obscured or hidden functionality can be missed.
- ▶ Logical design flaws cannot be detected easily.

An example of black box analysis is running a Web application scanner against a deployed application in the delivery phase.

Gray box analysis

Both white box and black box analysis can reveal possible software risks and potential exploits. White box analysis has guaranteed code coverage but a hard-to-measure risk. Black box analysis guarantees the ability to identify real problems but at the cost of unknown code coverage. The use of gray box techniques combines both white and black box analysis methods in a powerful way. The benefits of both approaches can be obtained while minimizing the potential that important issues are missed.

The challenge of deploying gray box analysis is that it usually requires correlating the result sets that are obtained in different phases of the software development life cycle. It can be difficult to adopt a successful gray box analysis strategy without a corporate culture shift. The highest chances of success are realized when gray box analysis becomes a fundamental part of the software development life cycle, allowing for smooth integration of different test results obtained throughout the life cycle.

Note: A more common way to use these terms is to differentiate between source code access (white box) and no source code access (black box). While this is not accurate in the purist sense, it is often the popular nomenclature.

Using the security analysis techniques in your development

When looking at the broader picture of the software development life cycle, we can see that each of the security analysis techniques have their function in the software development life cycle. For complete coverage, you must combine white box and black box testing into one security testing solution as part of the software development life cycle.

Figure 4 shows how architectural analysis, source code analysis, dynamic analysis, binary analysis, and vulnerability scanning can be used and combined throughout the design, development, and delivery phases. It is important to realize that security testing is *not* something that can be done only at the end of your software development life cycle just before the product release. Although a thorough security audit during the delivery phase is an important aspect in securing your software properly, security must be integrated into every step of the development life cycle.

Architectural security analysis should be a part of your design phase. Source code analysis should be integrated into your development phase. Dynamic analysis starts in the development phase and lasts all the way to your delivery phase. Binary analysis is something that you typically do once during the audit part of the delivery phase for each new build of your software. Finally vulnerability scanning is a repeating task that should be done periodically as long as your software is operational.

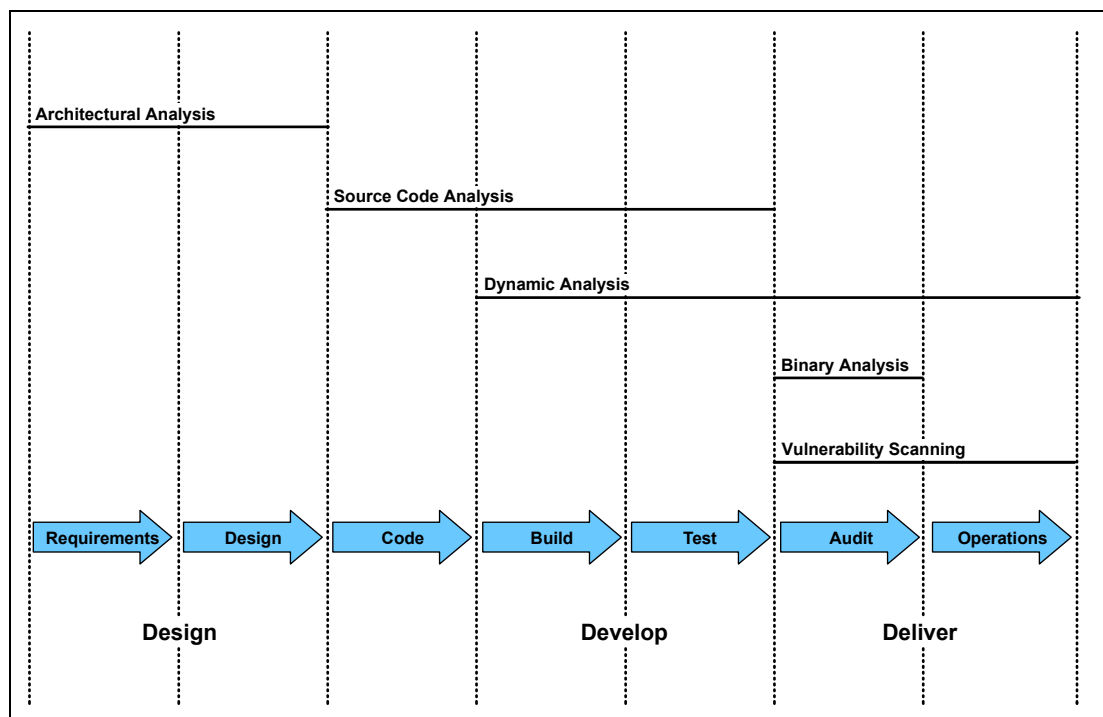


Figure 4 The different types of security analysis throughout the software development life cycle

Two important best practices

Let us take a closer look at two of the important best practices:

- ▶ Defining clear security requirements
- ▶ Maximizing automatic testing of those requirements

Defining clear security requirements

To clearly define security requirements, it is important to know exactly what a security requirement is. Despite the fact that they serve a fundamentally different purpose, security requirements are often confused with security features. The difference can be made clear with the following two definitions:

- ▶ *Security requirements* are *tasks* that *must* be done. For example, all passwords must be stored as encrypted. A security requirement typically does not indicate how this is done, such as indicating which software to use to do the encryption.

- ▶ *Security features are functionalities that must be available.* For example, account management must be possible through a Web interface.

Similar to functional requirements or performance requirements, security requirements are needed to ensure that security is built into the application from the start. Security requirements define what new security features are required and how existing features should be changed to include necessary security properties. The objective of security requirements is to ensure that the application can defend itself from attacks. You must consider the nine categories of security requirements, as explained in the following sections, when building a Web application.

Auditing and logging

While people often depend upon network and packet logs for forensics analysis, it is equally important that an application has an internal logging ability for events that are important to the confidentiality, availability, and integrity of the software.

For example, the application requires an audit log. Logged events must include the current session token (if available), IP address, and time. The events that must be logged include account authentication attempts, account lockouts, application errors, and input values that do not match the specified validation routines.

Authentication

Because most applications have access control restrictions to prevent confidentiality breaches, it is important that these access control mechanisms cannot be reverse engineered or manipulated to allow unauthorized access.

For example, strong passwords are required. Any authentication credentials must contain the appropriate strength, including uppercase, lowercase, and numeric characters, and be no fewer than eight characters in length.

Session management

The HTTP protocol was not originally engineered in a way that easily allows a session to be tracked over the duration of an application session. This led to session management capabilities to be built on top of the HTTP protocol.

An example requirement might be that the application must remain available to legitimate users. All resource utilization by remote sessions must be monitored and limited to prevent or mitigate attacks on application availability.

Input validation and output encoding

While most design-level security flaws are discovered and mitigated during the modeling and architecture phases, most development and delivery security issues are introduced because of poor input validation and output encoding. It is critical that any user-supplied data goes through the appropriate validations.

For example, all input must be validated through a central validation control.

Exception management

In the purist sense, it is impossible for an application to be completely secure. However, it is most often acceptable to be “secure enough.” Hiding detailed application exceptions or over-specific error messages can go a long way toward increasing the time required to compromise an application.

As an example, a security requirement in this context is that all error messages should be trapped and logged in the secure audit log.

Cryptography

Secure cryptographic algorithms are notoriously difficult to create and implement. It is extremely important that an organization choose the industry supported algorithm that meets the need of the business.

An example security requirement might be that all cryptography algorithms used within the application must be Federal Information Processing Standards (FIPS) [17] approved and compliant.

Data at rest

While all applications attempt to secure the data in the back-end repository, it is best to assume that, at some point, this data store will be leaked. Defense in depth dictates that any sensitive data be encrypted for this eventuality.

One requirement might be that, if the application contains sensitive user information that must be protected for regulatory compliance, strong encryption must be used to protect the sensitive user information such as name, user name, address, and financial data.

Data in motion

As long as software applications are implemented, there are known attacks against the software architectures that require transmission of data across networks or systems. It is critical that all application data be properly protected as it crosses both open and closed networks and systems.

A security requirement for such a system might be that, if the application transmits sensitive user information over untrusted or unsecured networks, then all communication channels must be encrypted.

Configuration management

New vulnerabilities appear daily, highlighting weaknesses in common infrastructure components. While some of these issues can be corrected through patching, it is sometimes a requirement that these options be available to the user for specific deployments. It is critical that the underlying systems be balanced in such a way that it meets the application business need, while protecting the application and infrastructure.

An example security requirement is that administrative interfaces must be separate from non-administrative interfaces.

Maximizing automatic testing of those requirements

After you define a clear set of security requirements, it is important to have a solid process in place to check if these requirements are correctly used throughout your development life cycle. If you think back about the motivations of attackers and the way they operate, it comes to mind that they try to cut costs by automating attacks on your Web applications.

In the same way as your attackers automate their tasks, is possible for you to automate the testing of your security requirements and verify the correctness of their implementation. In fact, if you make sure that vulnerabilities that are easily found in an automated fashion are no longer present in your Web applications, you will have eliminated a high percentage of possible attacks.

Therefore, it is a good practice to make sure that you use automated testing to secure your Web applications. Since automated testing cannot cover all vulnerabilities, if necessary, this testing can be augmented with manual ethical hacking and manual code revisions based on your business requirements. In the next section, we show you how you can use IBM Rational AppScan products to automate security testing in your development life cycle.

The IBM Rational AppScan software suite

IBM Rational AppScan is a suite of Web application security testing products that are used to automate application scanning and vulnerability identification. The Rational AppScan products scan and test for a wide range of Web application vulnerabilities, including those identified by the Web Application Security Consortium (WASC) threat classification and the Open Web Application Security Project. The Rational AppScan product line contains a wide variety of products, each of them adapted to the needs of a specific user.

In the remainder of this section, we discuss the main editions in order of their appearance in the software development life cycle. For more information about the full set of Rational AppScan products, see the following address:

<http://www.ibm.com/software/awdtools/appscan/>

Rational AppScan Developer Edition

The first user targeted by the Rational AppScan product line is the developer. The most efficient way to stay ahead of application security vulnerabilities is to build software securely from the ground up. The challenge is that most developers are not security experts, and writing secure code is not always their top priority. Therefore, the best way to engage development in the process of application security is to provide them with tools that work in their environment and that generate results in languages they understand.

IBM Rational AppScan Developer Edition is designed to empower developers to invoke Web application security testing from within their development environment. It enables addressing the volume of security issues that can be introduced in code at an early point, streamlining the development life cycle workflow, and helping to reduce security testing bottlenecks that can occur at the end of the release cycle. Rational AppScan Developer Edition uses a range of analysis techniques to accurately pinpoint security issues in your Web applications, including static code analysis, dynamic analysis, run-time analysis, and string analysis.

When it comes to security testing, the needs of the developer are quite different from those of, for example, the security auditor. Rational AppScan Developer Edition is meant to be used as a development tool. Therefore, its focus is on ease of use and ease of integration into the development process. The features to minimize false positives and provide easy-to-understand results are given higher priority than those to increase the breadth of a scan, which can complicate security testing. Collaboration and sharing of configurations and results are a core part of the product, and reuse of a scan configuration helps provide consistent, repeatable scans on each application.

The developer edition is designed with automation in mind. Automating code analysis configuration for greater ease of use and accuracy is realized by using *string analysis*, which is a new technology invented in collaboration with IBM Research. String analysis presents a breakthrough in the field of static code analysis by helping to solve the biggest challenge that plagues current security code scanning solutions, which is false positives. Until now, the most advanced technique for analyzing code security, taint analysis, has tracked input values as they flow through the system. However, it relies on the developer to determine whether the data is properly sanitized by marking the sanitization functions.

Therefore, the developer must know what foolproof sanitization is and must be able to configure it in detail for the analyzer tool to be accurate. These inherent limitations result in an overwhelming number of false positives, a frequent need to modify code to support the scanning tools, and the need to involve a security expert. String analysis automates these decisions, helping to eliminate false positives and supporting the different methods of handling inputs. While taint analysis measures whether an input is tainted, string analysis can

determine exactly how the input is tainted, bringing static code analysis to new levels of accuracy.

In addition to the self-serve benefit of string analysis, Rational AppScan Developer Edition provides built-in training, accurate and prioritized results pointing straight to questionable lines of code, and detailed remediation advice with code samples. These intuitive, easy-to-use features allow developers to be self-sufficient in their daily handling of Web application security testing.

Figure 5 shows a Rational AppScan Developer Edition scan report, illustrating how it links a white box issue and the affected line of code, allowing quick location and easy mitigation of the problem. Rational AppScan Developer Edition also has various capabilities of integrating with other products that are illustrated in the next section.

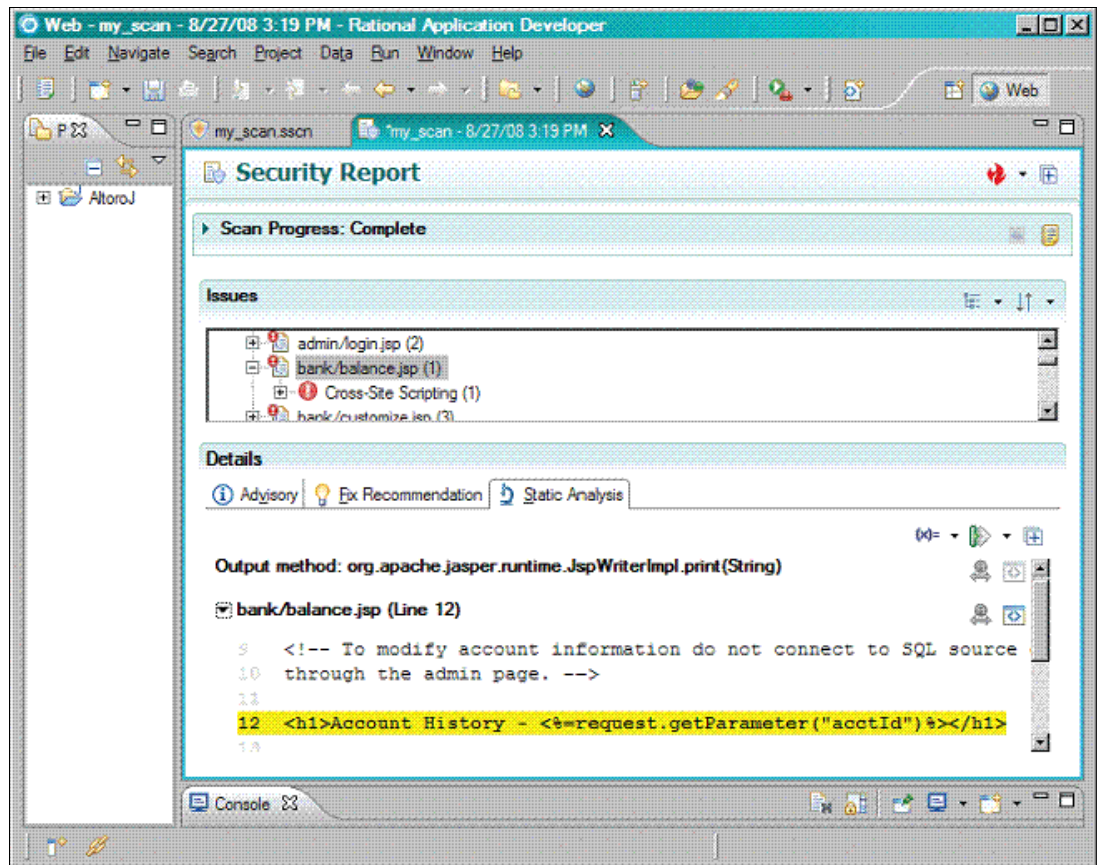


Figure 5 Completed scan report in Rational AppScan Developer Edition, with a white-box issue and the affected line of code

For more information about Rational AppScan Developer Edition, see the following Web address:

<http://www.ibm.com/software/awdtools/appscan/developer/>

Rational AppScan Build Edition

An additional way to use the vulnerability detection capabilities of Rational AppScan Developer Edition vulnerability scanning engines is to perform automated scanning during the builds. Integrating security into the build phase can be done by using IBM Rational AppScan Build Edition. By integrating with multiple build management systems, such as IBM Rational Build Forge® software, Rational AppScan Build Edition provides security testing coverage for scheduled builds. It includes the same set of analysis techniques as the Rational AppScan

Developer Edition and provides a high level of accuracy plus code coverage that helps you identify which code has been tested.

After scanning, Rational AppScan Build Edition routes the results back to development through defect-tracking solutions, such as IBM Rational ClearQuest® software, or through security reporting solutions, such as Rational AppScan Enterprise Edition or Rational AppScan Reporting Console. Rational AppScan Build Edition also includes an application programming interface (API) and various other result formats to support the propagation of scan results to other repositories.

For more information about Rational AppScan Build Edition, see the following Web address:

<http://www.ibm.com/software/awdtools/appscan/build/>

Rational AppScan Standard Edition

The next user that IBM Rational AppScan products consider is the security auditor. To help this user, the IBM Rational AppScan Standard Edition was created. To allow the security auditor to automate testing of the latest technologies, Rational AppScan Standard Edition supports the latest Web 2.0 technologies; parsing and execution of JavaScript™ and Adobe® Flash applications; asynchronous JavaScript and XML (AJAX) and Adobe Flex-related protocols, such as JavaScript Object Notation (JSON), Action Message Format (AMF), and SOAP; elaborate service-oriented architecture (SOA) environments; and custom configuration and reporting capabilities for mashups and process-driven applications.

By automating many repetitive tasks, Rational AppScan Standard Edition reduces the costs that are associated with manual vulnerability testing. Whether you outsource your vulnerability testing or perform it manually in house, Rational AppScan Standard Edition dramatically reduces the time needed to perform a comprehensive vulnerability assessment of your applications. This enables you to evaluate your Web security posture on an ongoing basis, as opposed to performing quarterly or yearly audits, resulting in enhanced security levels and controllable costs.

The Rational AppScan Standard Edition scanning engine provides high levels of scan accuracy and dramatically limits false positives. To further improve accuracy and performance, it includes an adaptive test process that intelligently mimics human logic to adapt the testing phase to individual applications. Rational AppScan Standard Edition learns the application, down to the level of each specific parameter, and adjusts to perform only the tests that are relevant. To help ensure protection from the latest threats, Rational AppScan Standard Edition checks for attack signature updates from the IBM team of security research experts each time the software is launched.

One of the most critical aspects of Web vulnerability scanning is the quick remediation of issues. Rational AppScan Standard Edition provides a fully prioritized list of the vulnerabilities found with each scan, which enables high-priority problems to be fixed first, helping organizations to focus on what matters the most from a security perspective. Each vulnerability result includes a full description of how the vulnerability works and the potential causes. Integrated Web-based training provides short training modules directly from the user interface. The software's remediation view then explains the steps that are required to remediate the issue, including examples of both secure and insecure code.

Figure 6 shows a window from the Rational AppScan Standard Edition application.

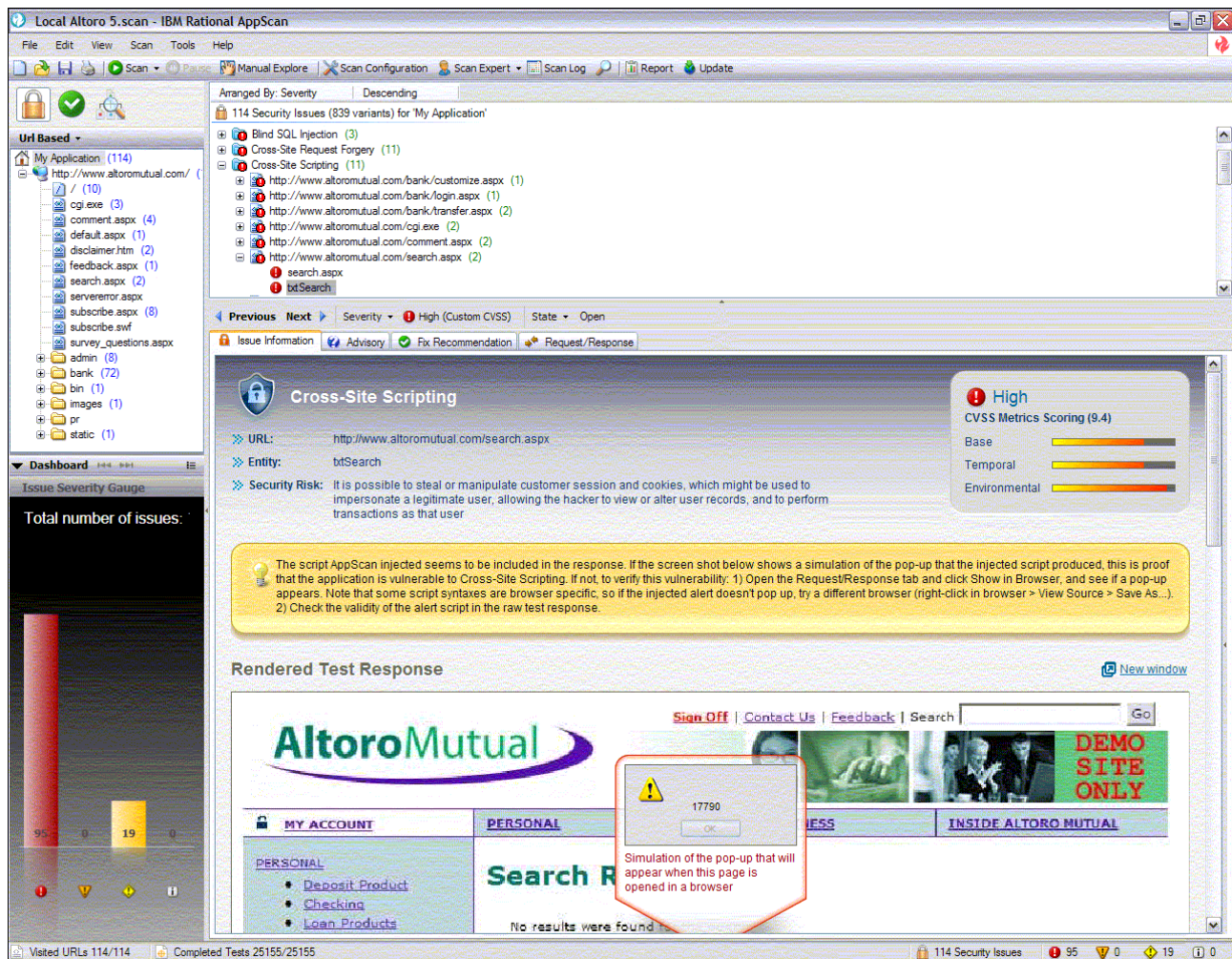


Figure 6 Rational AppScan Standard Edition helping users to quickly identify, understand, prioritize, and fix critical Web vulnerabilities

Rational AppScan Standard Edition can also help organizations address critical compliance requirements, such as Payment Card Industry Data Security Standard (PCI DSS), by providing a way to support an ongoing level of application security. IBM is an approved scanning vendor (ASV) with its Rational AppScan Standard Edition offering, making the software a perfect choice for addressing application security requirements around PCI DSS.

Rational AppScan Standard Edition can produce custom security reports and includes the ability to select which data points to include in each report. Users can also choose from more than 40 predefined reports and map scan results to key industry and regulatory compliance standards. These include National Institute of Standards and Technology Special Publication (NIST SP) 800-53 and the Open Web Application Security Project top 10, PCI DSS, Sarbanes-Oxley, Gramm-Leach-Bliley Act (GLBA), Health Insurance Portability and Accountability Act (HIPAA), Family Educational Rights and Privacy Act (FERPA), Freedom of Information and Protection of Privacy Act (FIPPA), and Payment Application Best Practices (PABP).

To customize and extend your testing for greater control, Rational AppScan Standard Edition includes a set of powerful customization features. The IBM Rational AppScan software development kit (SDK) offers a powerful set of interfaces that enable customizable invocation of each action in Rational AppScan Standard Edition, from the execution of a long scan to the

submission of an individual custom test. This platform enables easy integration into existing systems, supports advanced custom uses of the Rational AppScan engine, and provides the foundation for the Rational AppScan eXtensions Framework and Pyscan.

By automating Web application testing processes to help security auditors and penetration testers quickly and efficiently do their jobs, Rational AppScan Standard Edition, which is available as a desktop application or as *Software as a Service* (SaaS), assists in significantly raising the cost of attacks for your attackers, and therefore, no longer making your organization a valuable target.

For more information about Rational AppScan Standard Edition, see the following Web address:

<http://www.ibm.com/software/awdtools/appscan/standard/>

Rational AppScan Tester Edition

Let us take one step back from the delivery process into the development process and look at how we can integrate Rational AppScan products into the test phase. By using the same vulnerability detection capabilities as Rational AppScan Standard Edition, IBM Rational AppScan Tester Edition, which is available as a desktop application, offers capabilities to help quality assurance (QA) teams integrate security testing into existing quality management processes, thereby easing the burden on security professionals. Because Rational AppScan Tester Edition integrates with leading testing systems, QA professionals can use its functionality in test scripts. They can conduct security checks within their familiar testing environments, facilitating the adoption of security testing along with functional and performance testing.

Given that QA organizations already know how to prioritize defects, make testing repeatable, and report on test coverage and release readiness, these teams are ideally suited to perform security testing. They can help scale security testing and find and remediate security vulnerabilities earlier in the application delivery process. They can do this while they are testing for functional and performance issues and prevent costly delays in your releases.

For more information about Rational AppScan Tester Edition, see the following Web address:

<http://www.ibm.com/software/awdtools/appscan/tester/>

Rational AppScan Enterprise Edition

IBM Rational AppScan Enterprise Edition is a Web-based, multi-user Web application vulnerability testing and reporting solution for organizations that need to scale application scanning across the organization while maintaining centralized control of vulnerability data. Rational AppScan Enterprise Edition includes QuickScan, a point-and-shoot testing tool, and integrated Computer Based Training to accelerate the adoption of security testing across the software development life cycle.

In addition to its advanced application scanning, Rational AppScan Enterprise Edition provides sophisticated reporting and remediation capabilities and seamless integration with the desktop version of Rational AppScan, Rational AppScan Standard Edition. Rational AppScan Enterprise Edition helps users understand their overall security posture by generating executive security metrics, dashboards, and key regulatory compliance reports.

Rational AppScan's scan engine operates by traversing a Web application, analyzing and testing the application for security and compliance issues, and generating actionable reports with fix recommendations to simplify the remediation process. These advanced fix recommendations deliver unmatched accuracy and efficiencies for developers and security auditors to help address and remediate vulnerabilities disclosed by the scan. Seamless integration with leading QA testing tools (including IBM Rational ClearQuest), development

environments, and code scanning devices further simplifies security testing and remediation by QA and development teams.

With its Web-based architecture, Rational AppScan Enterprise Edition is designed to help organizations distribute responsibility for security testing among multiple stakeholders. Rational AppScan Enterprise Edition is for teams who need to perform Web application security assessments in a centralized fashion and provides a fully integrated solution set. Figure 7 shows the Rational AppScan Enterprise Edition dashboard.

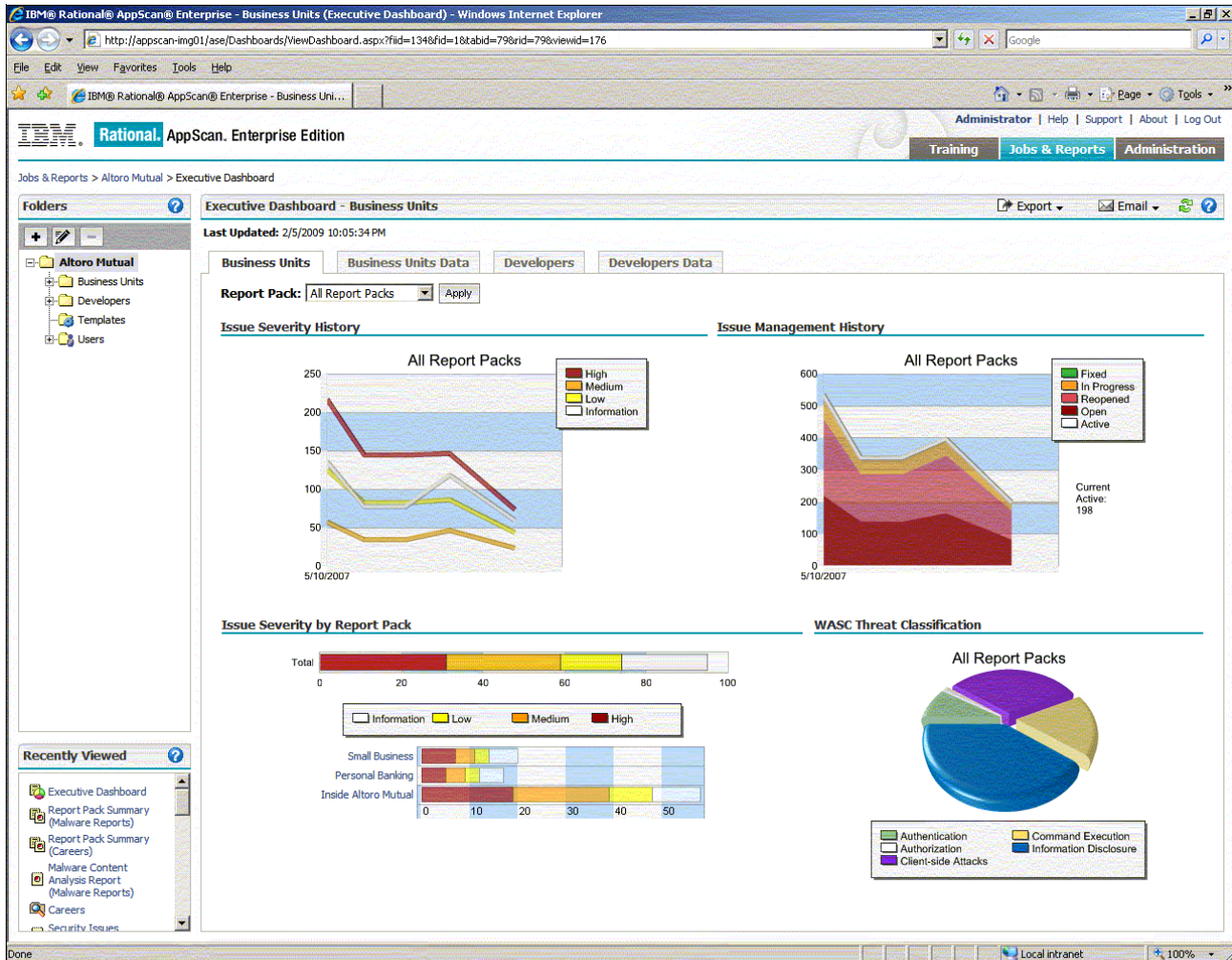


Figure 7 The IBM Rational AppScan Enterprise Edition dashboard view

For more information Rational AppScan Enterprise Edition, see the following Web address:

<http://www.ibm.com/software/awdtools/appscan/enterprise/>

Combining IBM Rational technologies into a comprehensive, integrated secure platform

From the previous section, you can see that the Rational AppScan product line by itself does not cover the full software development life cycle. To create a comprehensive, integrated secure development platform, Rational AppScan products are to be integrated with additional products. For example, at this point, we have not covered the requirements and design

phases, nor did we talk much about the products to track and report the vulnerabilities that were found by the AppScan products.

In this section, we show how other IBM products can be integrated with the Rational AppScan product suite to create a fully secure development life cycle. While we present IBM Rational software products, it is also true that competitive technologies help to deliver the same end goal. Because not all technologies are created equal, we start with grouping the technologies into four tiers of importance.

The first technology tier required for a secure software development life cycle should not be surprising: an integrated development environment, with source control and change request management. Almost every development shop recognizes the importance of code provisioning and tracking defects. Without a central repository to manage the code and defects, the building of software might be a completely ad hoc process. Such technologies as Rational Application Developer for WebSphere® Software, Rational ClearQuest, and Rational ClearCase® establish a baseline for repeatable, measurable software creation.

- ▶ *IBM Rational Application Developer for WebSphere Software* is designed to help developers quickly build high-quality Java™; Java Platform, Enterprise Edition (Java EE); Web; Web services; portal; and SOA solutions. The integrated development environment (IDE) assists in rapidly designing, developing, assembling, testing, and deploying these applications. The software's visual tools help reduce manual coding by abstracting the Java EE programming model. They make it easier and faster to complete development projects and let you focus your efforts on creative software solutions.

For more information about Rational Application Developer for WebSphere Software, see the following Web address:

<http://www.ibm.com/software/awdtools/developer/application/>

- ▶ *IBM Rational ClearCase* is an industry-leading solution that provides sophisticated version control, workspace management, parallel development support, and build auditing to improve productivity. This comprehensive software configuration management offering delivers a strengthened, centralized deployment model that makes it even easier for global teams to coordinate efforts. As development teams continue to face greater pressure to deliver higher quality software faster than ever, Rational ClearCase can help simplify the software delivery process and boost productivity.

For more information about Rational ClearCase, see the following Web address:

<http://www.ibm.com/software/awdtools/clearcase/>

- ▶ *IBM Rational ClearQuest* provides change tracking, process automation, reporting, and life cycle traceability for better visibility and control of the software development life cycle. It is designed to help you manage the software life cycle more effectively. It gives you access to the information you need to make better decisions. It helps you to more effectively manage tasks and schedules, and respond rapidly to customer needs. The automated workflows of Rational ClearQuest can help you control and enforce development processes and help improve team communication, productivity, and quality.

For more information about Rational ClearQuest, see the following Web address:

<http://www.ibm.com/software/awdtools/clearquest/>

The second tier of components deals with two additional categories: requirements and test management. A product such as Rational RequisitePro® allows business professionals and architects define explicit, measurable requirements within the software. The automated security testing products of the Rational AppScan product line allow automation to be effectively used at different software development life cycle phases to test for defined security vulnerabilities. All of this combined with a product such as Rational Quality Manager, which collects the information to one place, creates a system that tracks tests, test results, and

defects, as well as answers that last and final big question: Are we ready to ship? These top two tiers are the absolute minimum for delivering secure software.

- ▶ *IBM Rational RequisitePro* is designed for project teams who want to manage their requirements, write good use cases, improve traceability, strengthen collaboration, reduce project risk, and increase quality. It provides a scalable and fast Web interface for distributed teams and an enhanced, security-rich model for enterprise deployments. In doing so, it enables customized requirements access for business analysts, architects, designers, developers, and testers allowing integration in many phases of the secure software development life cycle.

For more information about Rational RequisitePro, see the following Web address:

<http://www.ibm.com/software/awdtools/reqpro/>

- ▶ *IBM Rational Quality Manager* is a Web-based centralized test management environment. It is for business, system, and IT decision makers and quality professionals who seek a collaborative and customizable solution for test planning, workflow control, tracking, and metrics reporting capable of quantifying how project decisions and deliverables impact and align with business objectives. It is designed to help teams collaborate by allowing them to seamlessly share information, use automation to accelerate project schedules, and report on project metrics for informed release decisions.

For more information about Rational Quality Manager, see the following Web address:

<http://www.ibm.com/software/awdtools/rqm/>

In recent years, there has been a shift of development from the traditional “Big Design Up Front” waterfall methodology, to the more iterative and incremental agile methodology. This shift has allowed businesses to realize real cost savings by answering critical questions about the software success and viability early on. It also allows them to react much quicker to customer suggestions and demands connected with the software. However, it also means that many more builds are created.

A solution for this new way of building software is answered through a product such as Rational Build Forge. It allows the development team to maintain an automated build management system that provides instant feedback required for this new development methodology. The reason why this is considered an essential component for building secure software is that integration with Rational AppScan allows security analysis to be done iteratively on every build. It provides early and valuable feedback that reduces both costs and risk at the later stages of the software development life cycle.

- ▶ *IBM Rational Build Forge* is an adaptive process execution framework that automates, orchestrates, manages, and tracks all the processes between each hand-off within the assembly line of software development, creating an automated software factory. Rational Build Forge integrates into your current environment and supports major development languages, scripts, tools, and platforms. It allows you to continue to use your existing investments while adding valuable capabilities around process automation, acceleration, notification, and scheduling.

For more information about Rational Build Forge, see the following address:

<http://www.ibm.com/software/awdtools/buildforge/>

The next tier of components that directly relate to security is the architectural and asset management systems. Architectural management software, such as Rational Software Architect for WebSphere Software, allows development teams to quickly design and reuse proven secure designs that demonstrate secure interactions between components. Asset management systems, such as Rational Asset Manager, allow development teams to maintain a library of certified, secure components that developers can quickly turn to for common tasks, such as authentication, authorization, input validation, logging, auditing, and

so on, that if mistakenly implemented, can lead to system compromise. When this proven asset library does not meet the requirement and a custom component must be written, the security team can be called in to evaluate.

- ▶ *IBM Rational Software Architect for WebSphere Software* is a powerful, integrated design and development environment. It can help IT architects and developers understand, design, manage, and evolve solutions across the team, across the world, and across different areas of technical expertise. Its abstraction, analysis, and reporting capabilities are designed to enable more effective communication and collaboration. In addition, its automation and intelligent editing tools can help improve productivity, enhance architectural control, and ease the design-to-code experience for Java and Java EE, Web services, SOA, and Web 2.0 applications.

For more information about Rational Software Architect for WebSphere Software, see the following Web address:

<http://www.ibm.com/software/awdtools/swarchitect/websphere/>

- ▶ *IBM Rational Asset Manager* reduces software development costs and improves quality by facilitating the reuse of all types of software development related assets. It is a collaborative software development asset management solution that helps you locate, share, and trace assets across business and deployment teams. Rational Asset Manager allows organizationally distributed development teams to identify, manage, and govern the design, development, and consumption of software assets, including services as part of an SOA initiative. The software facilitates the development, deployment, and use of collaborative software assets, helping IT organizations deliver innovative IT solutions, while controlling costs, reducing application backlogs, and improving business flexibility and responsiveness.

For more information about Rational Asset Manager, see the following Web address:

<http://www.ibm.com/software/awdtools/ram/>

After you reach the stages of deployment within your operation environments, the last tier of products allows you to keep the delivered product and its data secure. IBM Optim™ Data Privacy Solution can ensure you that your client data is fully protected and masked in case of a leak. To automate the task of keeping your network within your enterprise safe, Proventia® Network Enterprise Scanner and Proventia Network MFS can be incorporated into your network infrastructure. To keep your servers and your desktops fully patched, IBM Tivoli® Security Compliance Manager provides a centralized solution to keep track of the security of your machines.

- ▶ *IBM Optim Data Privacy Solution* enables the safeguarding of the privacy of client data. De-identifying confidential data is one of the best ways to protect privacy and support compliance with regulations such as HIPAA, DDP, PIPEDA, PCI DSS, and others. Optim Data Privacy Solution delivers powerful data transformation capabilities to mask confidential corporate data, so that you can use it safely for application testing. You can safeguard vulnerable test environments by applying simple data masking techniques, or apply pre-packaged transformation algorithms for complex data elements such as credit card numbers, e-mail addresses, and national identifiers.

For more information about the Optim Data Privacy Solution, see the following Web address:

<http://www.ibm.com/software/data/data-management/optim/data-privacy-solution/>

- ▶ *IBM Proventia Network Enterprise Scanner* allows you to know what is on your network and where potential problems lie. This solution is essential to identifying and managing risk. Complying with security regulations and outlining remedies can be costly and labor intensive. Proventia Network Enterprise Scanner enables you to save both cost and time in managing your network vulnerabilities. Proventia Network Enterprise Scanner helps

ensure the availability of your revenue-producing services and protects your corporate data by identifying and prioritizing risks, assigning protection activities, and reporting on results.

For more information about Proventia Network Enterprise Scanner, see the following Web address:

<http://www.ibm.com/services/us/index.wss/offering/iss/a1027216>

- ▶ *IBM Proventia Network Multi-Function Security (MFS)* is a unified threat management (UTM) device that provides protection at the gateway and network levels without jeopardizing network bandwidth or availability. It combats a variety of threats at once, such as unauthorized access, network attacks, malicious code, blended threats, content-based attacks, spyware, and phishing. Proventia Network MFS integrates these best-of-breed security modules in a single high performance and easy-to-use UTM appliance: firewall/VPN, intrusion prevention, anti-virus, antispam, Web/URL filter, and application protection.

For more information about Proventia Network Multi-Function Security, see the following Web address:

<http://www.ibm.com/services/us/index.wss/offering/iss/a1027111>

- ▶ *IBM Tivoli Security Compliance Manager* identifies security vulnerabilities and security policy violations. It protects your business against vulnerable software configurations by defining consistent security policies and monitoring compliance of these defined security policies. It automates scans of servers and desktop systems, which can help reduce the cost and time that are associated with manual security checks.

Tivoli Security Compliance Manager provides reports to security officers and compliance auditors with detailed information about the security health of the business so that they can take the appropriate steps to make individual systems and departments compliant. It identifies software security vulnerabilities prior to costly damage being inflicted by security incidents, improves business operations, and helps to increase efficiencies through automation and centralization. Tivoli Security Compliance Manager assists in addressing compliance issues in regulations and standards by automating compliance tasks, monitoring correspondence, reducing human error, and taming compliance costs.

For more information about Tivoli Security Compliance Manager, see the following Web address:

<http://www.ibm.com/software/tivoli/products/security-compliance-mgr/>

Figure 8 shows an overview of how all of these technologies can be combined into one full secure development life cycle. By integrating security-related IBM products in each step of the development life cycle, the security of your final product can be significantly enhanced.

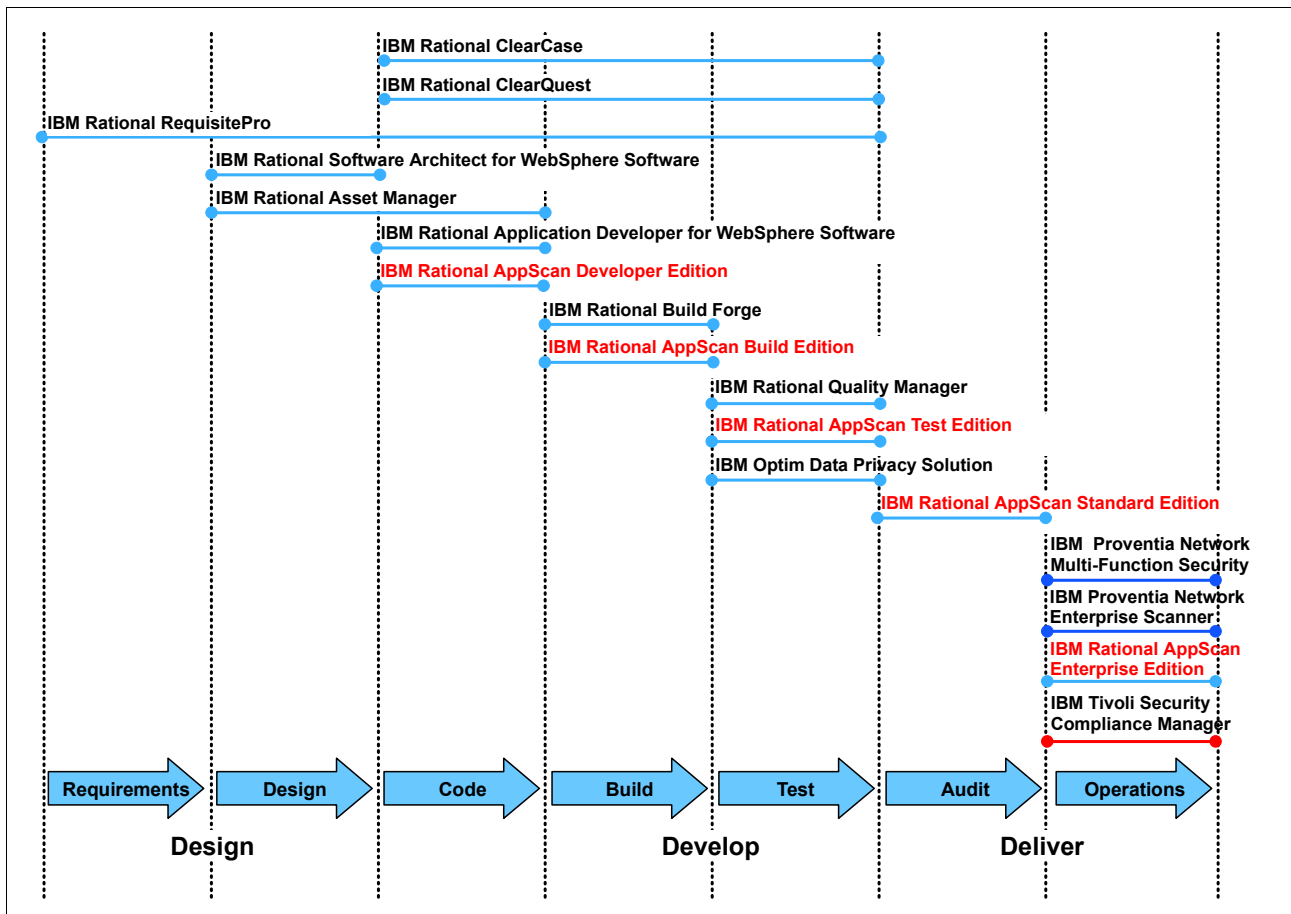


Figure 8 Combining IBM technologies to create a comprehensive, integrated secure platform

Business scenario: A step-by-step approach to Web application security

In this section, we look at a business scenario in which an organization that had no security testing becomes one that has state-of-the-art security testing fully integrated into its software development life cycle. In this scenario, after hearing about several security breaches at its bigger competitors, the fictitious company decided to look at IBM's products to improve their security to avoid the same negative publicity their competitors suffered from and to differentiate themselves by offering high quality secure Web-based products.

After talking to IBM, the company decided to turn their, currently insecure, software development life cycle step-by-step into a security-aware life cycle that produces high quality secure Web-based products. Because this company wanted to have immediate results, that is, they wanted to immediately know their security posture now, they initially chose to work with outsourced audits through Rational AppScan OnDemand Production Site Monitoring. This service allowed them to get immediate feedback while preparing to deeply integrate security into their systems. By doing this, they got access to the right resources and the right experts to further develop their security own plans. Parallel with this effort they began to

educate everyone involved in the development life cycle about security. This education was facilitated by the embedded Web-based training modules that help explain the vulnerabilities and demonstrate the exploits that were discovered by Rational AppScan OnDemand.

After the company trained a few of their own people to become sufficiently independent from the on-demand offering and able to perform their own security testing, they acquired the necessary licenses of the Rational AppScan Standard Edition and built their own in-house security audit team. By using the automatic scanning functionality of Rational AppScan Standard Edition, the in-house auditors discovered vulnerabilities at the end of the development life cycle. They reported them back to the developers for them to fix before the products were released.

Because of the continuing security training throughout the company, and the feedback of the in-house security audit team toward the developers, this company soon reached the point for the developers to take on more of the security responsibilities. To introduce security testing early in the development life cycle, the developers were provided with Rational AppScan Developer Edition. This allowed them to automatically discover vulnerabilities during coding and made them aware of bad, security related development practices that needed to be avoided. By catching vulnerabilities earlier in the development life cycle, the resources of the security audit team were freed up, allowing them to focus on more complicated vulnerabilities, thus further strengthening the security posture of the company.

Next more security testing was automated, and integrated into the build and test processes. Rational AppScan Build Edition and AppScan Test Edition were used to serve this purpose. At this point, the company had distributed security testing throughout their entire software development life cycle.

To give development and management a good view of how security was progressing as the development projects evolved, the company started using the Rational AppScan Reporting Console as a centralized point for security data collection. With Rational AppScan Reporting Console, they were able to have an overview of all the security reports that were created with the different software tools used in their software development life cycle to increase visibility and further strengthen their security.

Centralizing all the vulnerability scan reporting had prepared this company to grow while keeping their security under control. By marketing their security as a key differentiator to their competitors in the Web application space, they sparked the attention of the market and started to grow rapidly. This created the need to have a fully scalable solution for security testing throughout the company.

To cope with the size of their rapidly growing Web environment, the company introduced Rational AppScan Enterprise Edition into their software development life cycle. This scalable, enterprise architecture enabled them to keep scanning their massive amount of Web applications by using distributed scanning agents that they were able to control from one centralized reporting point in the Rational AppScan Enterprise Edition. This way they were able to continue to grow rapidly without having to compromise their marketable security.

Figure 9 shows an overview of how the company gradually integrated security into their software development life cycle to turn it into a rapidly growing company with state-of-the-art security testing.

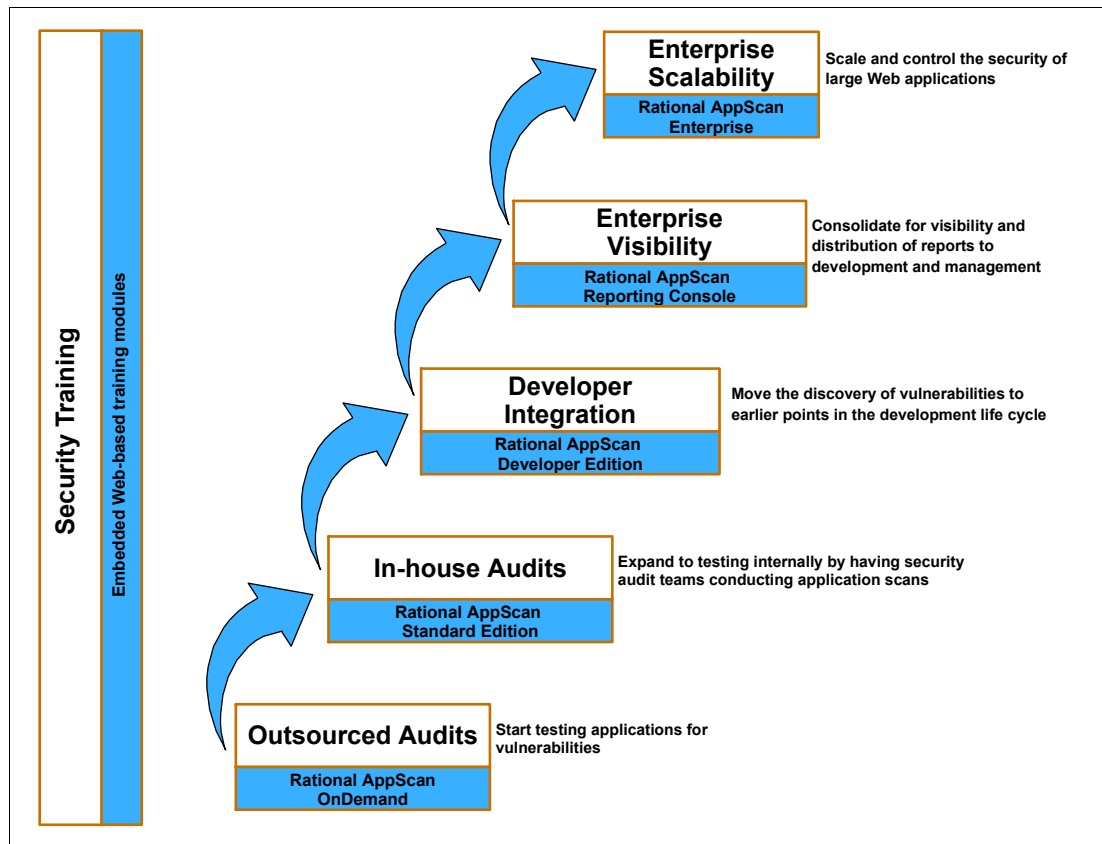


Figure 9 A step-by-step approach to Web application security

Summary

As we see an evolution on the Internet from hackers performing fame-hungry sabotage to fraud to profitable organized data and identity theft, business leaders must consider the security of their Web applications as a vital performance indicator of the success of their business.

In this IBM Redguide, looked at Web application security from an attackers perspective. By looking into the attackers' motivations and ways of operating, we demonstrated how they make money by attacking organizations on the Internet. We explained how attacking Web applications is a profitable business to them and how they use automation to cut costs and make more profit.

Next we showed you how, in the same way as they use automation to attack you, you can use automation in your software development life cycle to protect your organization. We introduced the IBM Rational AppScan product line and showed how you can integrate it into your software development life cycle to improve the overall security posture of your Web applications with state-of-the-art security testing.

Finally, we showed a scenario in which a company with no Web application security testing or knowledge, transformed into a security-aware company with state-of-the-art Web application

security testing fully integrated in their software development life cycle in a scalable and controllable manner. We showed how delivering high-quality secure Web applications allowed the company to differentiate itself from its competitors and strengthen its position in the market.

Other resources for more information

Consult the following sources, which have been referenced in this guide, for more information:

1. IBM Internet Security Systems X-Force 2008 Trend & Risk Report
<http://www.ibm.com/services/us/iss/xforce/trendreports/xforce-2008-annual-report.pdf>
2. Sharon Gaudin, "Estimates Put T.J. Maxx Security Fiasco At \$4.5 Billion," *InformationWeek*, May 2, 2007
<http://www.informationweek.com/news/security/showArticle.jhtml?articleID=199203277>
3. Nate Mook, "Cross-Site Scripting Worm Hits MySpace," *Betanews*, October 13, 2005
<http://www.betanews.com/article/CrossSite-Scripting-Worm-Hits-MySpace/1129232391>
4. Gary Warner, "Radical Muslim Hackers Declare CyberWar on Israel," *CyberCrime & Doing Time* blog, December 30, 2008
<http://garwarner.blogspot.com/2008/12/muslim-hackers-declare-cyberwar-on.html>
5. The MITRE Corporation, "Common Vulnerabilities and Exposures"
<http://cve.mitre.org/>
6. Dan Verton, "Airline Web sites seen as riddled with security holes," *ComputerWorld*, February 4, 2002
<http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=67973>
7. Web Application Security Consortium, "Web Application Security Statistics"
<http://www.webappsec.org/projects/statistics/>
8. Roi Saltzman and Adi Sharabani, *Active Man in the Middle Attacks: A Security Advisory*, IBM Rational Application Security Group, February 27, 2009
<http://blog.watchfire.com/AMitM.pdf>
9. Open Web Application Security Project (OWASP), "SQL Injection"
http://www.owasp.org/index.php/SQL_injection
10. Cisco Systems, "Understanding SQL Injection"
http://www.cisco.com/web/about/security/intelligence/sql_injection.html
11. Microsoft® SQL Server® 2008 Books Online, "SQL Injection"
<http://msdn.microsoft.com/en-us/library/ms161953.aspx>
12. RSnake, "XSS (Cross-Site Scripting) Cheat Sheet"
<http://hackers.org/xss.html>
13. Cgisecurity.com, "The Cross-Site Scripting (XSS) FAQ"
<http://www.cgisecurity.com/xss-faq.html>
14. OWASP
<http://www.owasp.org>

15. Web Application Security Consortium (home page)

<http://www.webappsec.org/>

16. Joris Evers, "Macworld crack offers VIP passes, hacker says," *CNET News*, January 12, 2007

http://news.cnet.com/2100-1002_3-6149994.html?part=rss&tag=2547-1_3-0-5&subj=news

17. Federal Information Processing Standards Publications (FIPS home page), Information Technology Laboratory

<http://www.itl.nist.gov/fipspubs/>

The team that wrote this paper

This paper was produced by a team of specialists from around the world for the International Technical Support Organization (ITSO).

Frederik De Keukelaere is a researcher at IBM Research, in the Tokyo Research Laboratory in Japan. He is part of the Security and Web Platform group where he is working on next-generation security models for the Web. His current research interest lies in usable Web security. He has over seven years of research experience in Web application security, Web, and multimedia technologies. He has actively participated in and contributed to various standards organizations such as the OpenAjax Alliance and the Moving Picture Experts Group (MPEG). He is the editor of several ISO/IEC standards and holds five ISO/IEC Awards for Outstanding Technical Contributions to MPEG-21. Prior to joining IBM in 2006, he was part of the Interdisciplinary institute for BroadBand Technology, in the Multimedia Lab in Belgium. During this period, he received a Ph.D. degree in Computer Science Engineering, from Ghent University, Belgium.

Danny Allan is director of security research with IBM Rational. Danny came to Rational through the acquisition of Web application security and compliance leader Watchfire in July 2007. He brings with him more than eight years of business and security technology-related experience, including penetration testing and internal system remediation for one of Canada's biggest universities. In his role as a security researcher, he is closely involved with enterprise global customer deployments, researching and evaluating technologies, and to helping define and recommend strategic directions. Danny has held several critical customer facing positions, including Team Lead, Consulting Services, and Sales Engineer. He has published several white papers and articles and participates in industry working groups. He has also spoken at security events and is often called upon by key media including the Associated Press, Bloomberg, and the Wall Street Journal for his opinions regarding Web application security. Danny holds a Bachelor of Commerce degree with a major in Information Systems from Carleton University.

Axel Buecker is a Certified Consulting Software IT Specialist at the ITSO in Austin, Texas. He writes extensively and teaches IBM classes worldwide on areas of software security architecture and network computing technologies. He holds a degree in computer science from the University of Bremen in Germany. He has 22 years of experience in a variety of areas related to workstation and systems management, network computing, and e-business solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in software security architecture.

Thanks to the following people for their contributions to this project:

Emma Jacobs, ITSO, IBM U.S.

Gary Vincent, IBM U.S.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document, REDP-4530-00, was created or updated on May 29, 2009.




Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>



The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AppScan®	Optim™	RequisitePro®
Build Forge®	Proventia®	Tivoli®
ClearCase®	Rational®	WebSphere®
ClearQuest®	Redbooks (logo)  ®	X-Force®
IBM®	Redguide™	

The following terms are trademarks of other companies:

Adobe Flash, Adobe, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Java, JavaScript, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, SQL Server, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.