

# Biggest Information Security Mistakes that Organizations Make

*and How to Avoid Making Them*

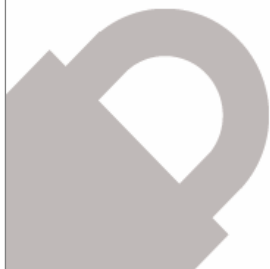
**Ed Adams**

CEO  
Security Innovation



PREPARED BY  
**SECURITYINNOVATION<sup>®</sup>**  
THE APPLICATION SECURITY COMPANY

Security Innovation, Inc.  
187 Ballardvale Street, Suite A170  
Wilmington, MA 01887  
+1.978.694.1008  
[www.securityinnovation.com](http://www.securityinnovation.com)



## Table of Contents

<b>Top Five Information Security Mistakes .....</b>	<b>3</b>
I. Over-relying on Network Defenses .....	3
II. Believing the Hype of Technology/Tools.....	5
III. Making too Many People Assumptions .....	6
IV. Assuming Secure Software is Costly.....	8
V. Falling into the "Recency" Trap .....	11
<b>Avoiding Mistakes: Six Good Practices .....</b>	<b>12</b>
I. Make a Self-Assessment .....	12
II. Believe the Application Security Hype .....	13
III. Ask Tough Questions .....	14
IV. Create an Internal "Red Team" of Ethical Hackers.....	14
V. Educate your Teams .....	15

Information security mistakes are costly, damaging and all too prevalent. Given the obvious repercussions of poor security strategies (see recent incidents from TJX, AOL, and the VA), one is inclined to believe change agents are in place; however, organizations continue to make seemingly avoidable mistakes when it comes to information security. This is due to misconceptions and common mistakes that are repeated.

This article introduces five common information security mistakes that organizations make and concludes with recommendations and best practices for building and maintaining a successful information security practice and avoiding these mistakes.

The first step toward enlightenment is knowledge - so here are the top five security mistakes organizations make, in no particular order:

1. Over-relying on Network Defenses
2. Believing the Hype of Technology/Tools
3. Making too Many "People" Assumptions
4. Assuming Secure Software is Costly
5. Falling into the "Recency" Trap

## Top Five Information Security Mistakes

### 1. *Over-relying on Network Defenses*

The problem isn't our networks (which are fairly well protected, btw), it's the crappy software we write and put on the network. There is no discipline or rigor to software engineering like there is in other engineering disciplines. I'm a mechanical engineer by trade so this is a very serious problem to me and one that I'm intimately familiar with. Examples in other industries are just as stark by comparison, e.g., doctors have residencies, civil engineers have to be certified and train under another certified engineer (they're called EIT, engineer-in-training) before they can lead a design projects, etc. There is no correlation in the software world and we, as organizations that build and buy software, aren't demanding a change. Network defenses like firewalls and intrusion prevention systems have a place in a multi-layered information security solution, but they can't protect us from the majority of vulnerabilities – those in the application layer. As a result, there are problems with these solutions that many organizations don't realize:

- ▶ they miss the majority of security vulnerabilities
- ▶ they give the user a false sense of security
- ▶ in many instances they *enable* damaging business logic attacks

### *Application vs. Network Security: Research/Analyst Perspectives*

Let's look at some research and analyst perspectives. Many of you are familiar with the Gartner statistic from 2004 (updated in 2006) that states: "Over 70% of security vulnerabilities exist at the application layer, not the network or system layer." NIST claims this number is 92%! IDC states, "The conclusion is unavoidable: any notion that security is a matter of simply protecting the network perimeter is hopelessly out of date." Another interesting metric that was collected from Microsoft Developer Research is: "64% of developers are not confident in their ability to write secure applications." That is a telling but disturbing sign - two out of every three developers in this survey were not confident in their abilities to write secure code. It's an interesting question you may want to ask your own developers. And while you're at it, ask this of your budgeting personnel: "If over 70% of security vulnerabilities exist

in the application layer vs. the network layer, are we spending over 70% of our IT security budget on application security?"

### **Ecommerce Case Study**

Following is case study from an e-commerce company that I worked with last year. This company had intrusion detection, intrusion prevention and a firewall in place. Because it was a large e-commerce site, we had to conduct testing on the production system. This e-commerce system had the common "shopping basket" functionality. During testing, we placed an item in our basket, did some testing, didn't find anything too unusual, so we closed the browser and went out for lunch. When we returned, we opened the browser and noticed that the item we had placed in our basket was still there – indicating that the e-commerce site used cookies. We decided to locate that cookie on our client and mess with it – something the security world calls "cookie poisoning." We opened the cookie with the world's ultimate hacking tool, Notepad, and found information like our session ID, the merchandise item number, a description of the item, and the price of the item. Hmm... price. We then decided to mess with that parameter and change the price from \$9.95 to *negative* (-\$9.95) and save the file with this new information. When we re-opened the browser, sure enough that item was now showing at -\$9.95. What a deal! We bought five. And shipping was calculated in the same manner, so we got that for a real bargain, too.

Now here's when the real trouble started. Since this was a live production site the order was actually placed. Because our systems are so compartmentalized today, the accounting department only got a message saying debit this account \$49.75. The people in shipping received a message saying to send five of these items to Security Innovation. There is no correlation or check between the processes. This order was never detected or blocked by the firewall or IDS because there was no abnormal behavior. The only thing our "attack" did was select an item, place it in the basket and check out. Of course, we cancelled the order before it actually shipped, but it was difficult. We had conversations with several teams before they could stop that shipment. This is a great example of why network security can give people a false sense of security and how we need to pay attention to the business processes that we think we're protecting. These network defenses enable business logic attacks because the watchers are looking the other way, thinking they're safe because they've got the latest and greatest deep-packet inspection firewall. Wrong.

Some buffer overflows can work the same way. Buffer overflows were *the* security bug of the 1990's – cross-site scripting and SQL injection soon took over as the high-profile threats this decade. But buffer overflows still wreak havoc on many systems because our network defenses don't have the context in which to understand well-crafted buffer overflow attacks. Take for example a string of data that comes over the network pipe. It may be part of a picture, it may be text, we don't know. But neither does the firewall watching the traffic. If this data happens to be part of a Flash, WMF, or PDF file, for example, the firewall has no way to determine if it is innocuous or evil. Firewalls have no context in which to understand how a piece of network traffic is going to be used by an application. In this example, an input buffer on a piece of freeware, e.g., Flash Player, Adobe Acrobat Reader, etc. can be overflowed and the client machine compromised very easily. Can't happen you say? This exact vulnerability existed for years until early 2006 in a ubiquitous piece of web software... and no network defense in existence could stop it from being exploited.

### **TCO Calculations**

Do you account for down-time and productivity losses when making your TCO calculations on freeware like Acrobat Reader and Flash Player? Or do you just assume your network defenses protect you? Think again. Network security defenses have a place in your security portfolio. They capture some malicious

users and are effective and stopping known attacks and viruses. But beware of the shortcomings so you aren't caught asleep at the wheel.

## **2. Believing the Hype of Technology/Tools**

Tools alone don't make people smarter, nor do they improve the process through which solutions are built. They simply make people and processes more efficient in jobs they are trained to do.

This is not to be meant to be a bash on security tools – my consulting company partners with security tool vendors and we believe they are valuable components to a mature security posture. But here are a few things to remember:

### ***Tools Don't Make People Smarter***

Tools don't teach a surgeon how to operate or a road worker how to jack-hammer a hole. I didn't become a better mechanical design engineer because I learned how to use AutoCAD; it just made me more efficient in the job I was already trained to do. This is especially true of Application Security tools; the market here is still nascent and users need more education before tools can be truly useful. Network security tools are similar but luckily the market, purpose, and limitations of the tools here are better understood and much more mature.

When using a scanner (source code or Web) remember that they are just that - scanners. They will create false positives and they will often miss serious vulnerabilities that do exist. Scanners are also flawed at catching business logic vulnerabilities, often the most damaging of all. These are vulnerabilities that exploit acceptable behavior to steal or circumvent checks in your system. Take the example of a negative integer attack on an ecommerce site. If the site uses client-side validation (many sites do, mainly for performance reasons), it's easy to poison a cookie and turn the price of an item from positive to negative. And because our business systems are so compartmentalized, most of these attacks go un-noticed. The accounting group receives a notice to debit this person's account -\$74.99, while the shipping department gets a message that instructs them to ship the item to the user; there is often no correlation between business functions and tools can't help.

### ***Tools need to be worked into your risk management & audit management cycles***

This is something consulting companies can help you do if you do not know how. There is a large translation problem right now in a lot of organizations where a risk management team defines a problem and must translate that to actionable activities for the software development and network operations teams that are going to be developing, deploying and managing the information systems. You need to take a three-pronged approach and integrate tools with your processes and your training. This helps your staff understand what's expected of them and provides the right tools to help them accomplish the job at hand - while adhering to both corporate and industry processes and regulations. Many organizations simply insert a tool into the software development or deployment process and require that an application "pass" some arbitrary score that has been predetermined. This is dangerous in both context and user interpretation. If an application passes, there may be a false sense of security that the application is secure. If the application fails, the development team may have more incentive to code specifically to pass the test rather than meeting critical business or security requirements. Tools and people are both fallible; thus, well documented and reviewed security policies/procedures are paramount.

There are a lot of good tools available but you need to make sure you give your team the context in which to use them as well as the training they need to get value from them. If you gave that jack hammer to a six year-old boy he's not going to know what to do with it or how to use it and two things are going to happen: he's either not going to use it at all because he's afraid of it, making it a very

expensive wall decoration, or he will use it and cause serious damage to him or someone else in the process. This is how a lot of companies use security tools today. They purchase the development or audit team a scanner and tell them to “just use it” without the guidelines or training on how to. What the organization needs to do is roll out sound processes integrating security into every step of the development and deployment process.

Intrusion Detection and Prevention Systems are classic examples of tools that give a false sense of security. They typically look for abnormal or suspicious data or behavior and then block that user or activity. Sounds great on the surface, but you need to understand exactly what they are looking at. Often, it’s no more than sniffing network traffic and watching for “out of bounds” behavior. I can’t tell you how many clients I’ve had that are frustrated because their IPS keeps preventing their users from making off-hour database queries or even a simple thing like flushing the browser’s cache. A rapid succession of deleting large numbers of files is often an activity that will get detected and blocked by an IPS or heuristic antivirus system.

### *I do Love Tools – I Swear*

I worked for a software testing tools vendor for 5+ years. But I also recognize that tools alone don’t make people smarter, or necessarily solve the problem you expect them to. That’s the short-term problem to focus on – train and educate your teams in the application development discipline and hold your network teams accountable to maintaining secure infrastructures.

Most of all, make sure your policies and procedures are up-to-date and include all use cases and abuse cases you need them to in order to protect your crucial assets. The best attempt I’ve seen to date is the PCI standard (Payment Card Industry) which is a specific and prescriptive set of requirements for secure information flow. This is not comprehensive, but it is getting there – for example, next year, it will require the presence of either a web application security firewall or proof that your application has been security code-reviewed. These are not replacements for each other, mind you, but it’s a good start in the right direction – a proactive attempt to secure data at both ends.

Security tools are not the panacea people want them to be. Just like network defenses, they have their place in a complete information security workflow, and they require people who know how to use them to be effective.

### **3. Making too Many People Assumptions**

Warning: I’m not taking the “glass half full” perspective here. This common pitfall is best described in a more curmudgeonly tone where we discuss psychological and judgment mistakes that are extremely detrimental and often overlooked in Information Security – too many people assumptions.

First, let’s agree that information security management is difficult. It is fraught with many unknowns and empty promises from technology vendors pledging to solve your security woes. These variables force you to make tough decisions, and often result in you extending inappropriate trust to your staff, employees, partners, and customers. For anyone who dealt with application performance issues circa 1998-2001, this is going to sound familiar. Information security is taking a rather similar approach to application and network performance to those few years ago that seem like a distant memory now (I think I’m just blocking out the pain.) It’s a difficult process where developers are not sure how to code or test for security, but management views it as part of their job and assumes they will figure it out, unaware that it’s a special (and mostly new) skill set and as such needs investment and time to develop in their teams. Meanwhile, hardware vendors are telling you to just throw some more iron or another appliance at the problem and you’ll be fine. Suddenly firewalls and IPS systems sound a lot like the network load balancing solutions of the late 90’s.

## ***Accept your State***

There is an absence of acknowledgement when it comes to information security – all too often, organizations assume that their network or IT staff has adequately protected them because they've got the latest anti-virus or firewall installed and running. This false sense of security is partly due to vendor misdirection and over-promising, but that's only a small part of it. People want to believe their information systems are secure, but consumers with that false sense of security are rather ignorant to some of the most dangerous threats that their systems face. Specifically, the dangers of insiders (both malicious and unintentional) and the sophistication of organized attacks on your information systems.

As much as I love compliance regulations because they keep the consulting world gainfully employed, too many a CIO has been duped into thinking that compliance means security; it does not. Take the data breach at Stop & Shop Supermarkets. This company is a poster child for PCI DSS (the Payment Card Industry's Data Security Standard). They are more secure than most companies I've worked with and yet they were still breached. Much to their credit, they disclosed right away and cooperated with the proper local and federal authorities leading the investigation (incidentally, there was an arrest in this case recently (see this [Boston Globe article](#)).

The Stop & Shop incident was more of a physical security issue than a digital one, unlike TJX Companies who made a series of people assumptions and got themselves into a heap of trouble as a result. First, they don't have a CISO role (someone solely responsible for information security); organizationally they do not place great importance on such a role. Further, they decided to hide the fact that they had incurred a loss of customer data instead of dealing with it directly and immediately. These bad choices ultimately led to their CEO resigning and two class action law suits: one from their customers and one from the banks who issued the credit cards. I suspect an FTC law suit is soon to follow.

## ***Education and Inertia***

Education is the next big people mistake organizations make. Not only are computer science graduates coming out of school with little or no information security know-how, but they are not getting security training on the job either – this is especially true with application security and development teams. We have noticed over the past few years working with our Fortune and Global 500 clients is that there is a HUGE demand for application security education -- everything from risk management teams to audit teams. This is a very encouraging trend, but the majority of companies still believe they don't need to educate their teams on information security.

Inertia is a huge culprit, too. There is always resistance to change particularly when you are limited to time and budget as most of us are – money is always the biggest driver. When making budget decisions, you also need to consider the risk of not making an investment. TJX might have been better served buying a web application firewall to protect themselves against common SQL Injection attacks instead of paying the bank of lawyers and incident response consultants who helped them make the decision to hide their security problems for years and break disclosure laws in the process.

## ***Know thy Enemy***

We all know there are malicious users, and they have become more sophisticated and more anonymous. A disturbing trend is the organized, targeted attacks on specific companies (see above). You've undoubtedly heard about NetBots. These nasty little pieces of ingenuity enable anyone to completely outsource an attack on a certain company, with complete anonymity. Insiders are a major threat, too, and often overlooked. We're not just talking about hackers anymore as a risk to companies, but people within the company who are both malicious and innocent. And who is an "insider" anyway? Key employees that work inside your building are obvious, but what about telecommuters? Are they an

insider just because they have login credentials and access to your network? Consultants? Temp workers? What about partners? If they are an insider, how much of an insider are they? Do they have access to all your strategies and pricing? Probably not. The line is fuzzy these days between insiders and outsiders and they all must be viewed as a threat to your business.

The causal hackers aren't the real threat. Many companies get uncomfortable with me saying so, but hackers actually help us! They trip land mines that are waiting to be exploited. You have much more control about your insider threat – so acknowledge it and act. Insiders already have access to your systems and know where the crown jewels are. A recent study done by the FBI crime lab reported a staggering statistic: over 80% of all computer crime was committed by insiders. Companies focus on hackers but that is the wrong assumption. And they always forget that it's their crappy software that allows the hackers to exploit them in the first place. These same defects are there for your partners, employees, and consultants to exploit too

### ***Two Troubling Case Studies***

One client of mine is an organization in the manufacturing sector. This company has an extranet where partners bid on parts, submit quotations, and respond to proposals. One partner was so paranoid about what their competitors were doing that they used a cross-site scripting defect in the extranet to escalate their rights to administrator level. Once they had uber-user privilege, they were able to view all bids and see exactly how competitors were pricing their products.

Another example involved a financial services company that outsourced its application development to a company in the Far East. This was a CMM level-5 company, which means it had a well established and documented process. This outsourcing company had a few malicious users on the payroll and they coded a back door in the application that was sent off to the client. This error was not caught immediately because the financial services company just did a cursory security scan as part of their acceptance testing. They made the fatal people assumption that their outsource vendor employed ethical staff and didn't do any checks on either the employees or the code they wrote. The back door was simply a URL that went undocumented and could be triggered remotely. Once the program was deployed, the malicious users were able to skim customer information such as account numbers, statement balances, and other information. The company who had deployed the system had no idea until their clients contacted them complaining of fraudulent charges. It was traced back to the embedded URL, but not until months later.

A company must be able to educate its employees on the risks facing it. These risks include writing applications securely, auditing outsourced functions for security holes, and providing training to everyone from your procurement team to your network IT staff. Unfortunately, we can't trust everyone implicitly. Background checks are useful, but secure your information systems from the inside out put you in a situation where you can't be burned as easily. Consider how to build in checks and balances to your critical data flow to protect yourself from bad people assumptions. Practice threat modeling and brainstorm with your management team about possible abuse case – you'll be amazed at what they come up with (and how unprotected you might actually be!)

And if you ever hear your team say, "We don't need any security training," or "It can't happen to us!" be afraid... be very afraid. Even the most skilled development teams in the world are green in some areas of IT and application security.

## **4. Assuming Secure Software is Costly**

Though it may add time to the up-front software development cycle,(i.e. defining requirements properly and designing systems well, integrating security into each phase of the SDLC, etc) designing for security saves a lot of time and money in later phases, especially testing and deployment when security holes

take a long time to troubleshoot, re-code, and patch. Microsoft and other organizations have some good case studies on utilizing their SDL (Secure Development Lifecycle) internally on applications and I'll touch upon one shortly.

### *You had me at Hello, World*

I recently had the pleasure of testing a very small application (225 logical source lines of code) that had several security vulnerabilities. Amazing as it sounds, this minimalist application was so poorly written that I thought it was written by a 10 year old (come to think of it, I know a few ten year olds who could write a better program!) And it got me to thinking how easy it is to screw up when writing software. That led to the (ahem) logical conclusion that we need to integrate security into our software at every possible phase... but is this feasible? Although it may add some time to the earlier phases of your software development cycle (SDLC), e.g., defining requirements and design, I contend that integrating security into each phase of the SDLC will ultimately save you a lot of time and money and reduce your overall TCO (total cost of ownership) for software you develop and/or deploy.

I have spoken with many clients who want to mandate security activities into their SDLC but are skeptical about the added time and complexity of doing so. They are worried about time-to-production protraction if they introduce new security activities for their application development and network teams. What I ask these clients is a simple question: "Have you considered the total cost and risks of both options, i.e., staying the course vs. introducing new security activities? Have you considered the cost of *not* making this security investment?"

### *Houston, we Have a Problem*

To start answering this question, we need some numbers to help us quantify. First, consider the research conducted by IDC and IBM that quantified the cost of fixing a defect versus discovery phase. Assuming a simple 4-phase approach (Design, Development, Test, Deploy), the cost of fixing a defect in development was 1/15th the cost of fixing the same defect if it's found post-deployment. The difference is even more severe when dealing with security defects because you have to factor in the likelihood of incident response costs, dealing with the media, customer disclosure/notification, and suddenly finding yourself out of compliance with a critical standard like PCI, SOX, or SB1386. Pile on top of that the difficult nature of troubleshooting security defects and typically longer times needed to re-code, test, and patch and you've got a pretty costly bug on your hands. You have to consider this as a risk and component of your total cost; if you don't, you're missing some potentially massive costs both short-term and long.

Next consider what Microsoft has done with products like SQL Server (developed using their SDL, Security Development Lifecycle process). I am no Microsoft apologist, but the data speaks for itself: SQL Server 2005 has substantially fewer security bugs than either the Oracle or MySQL databases that compete against it according to the CVE database. This translates into lower maintenance and support costs as well as improved public opinion and a competitive selling advantage. Will there be an additional short-term investment you need to commit to create secure software? Most likely, because you may need a little more training for your team and invest in some tools like team guidance systems and source code or web scanners to support your new activities. That initial cost versus your total remedial costs is really the metric you want to look at, not strictly the cost of creating secure code.

### *Show me the Money*

One company, an organization that specializes in electronic hand-held devices, was very driven by time-to-market. Since their products have a short life span they want to get to market as soon as possible but their CEO had recently gone public with a "commitment to security pledge" that helped boost their sagging stock price temporarily. In order to maintain the share price and honor the commitment to quality and security, they needed to create a time-to-market versus loss-of-market comparative analysis

where they went through the scenarios of their next gen hand-held device being released at a given date. They knew that every day shaved off their anticipated time-to-release schedule meant approximately an additional \$1,000,000 in profits to the business unit. But they had not considered the actual and realized costs if that product had a lot of security vulnerabilities that were discovered after release. Once they did that risk analysis, they realized that they would have very high remediation and patch costs, in fact, these costs dwarfed the million dollars/day savings they hoped to gain by shaving time off their product development lifecycle.

For the next release, they integrated security into their SDLC after sending 1/3 of their developers through training. The result was that it added 6 days to their release cycle (a “loss” of \$6,000,000) and they had 14 fewer high-severity security defects (a savings estimated at \$98,000,000 based on their historical costs.) Note, that \$98M number ignored any impact to stock price. In the following release of their product, they were only +1 in their time-to-market estimates and once again experienced a drop in number of security defects reports. An interesting side bar is that their developers were coding about 20% fewer security defects per release, but their development and test teams were finding 85% more defect pre-release. That is a fix-find rate any team would be happy with.

### *Second Verse Same as the First*

Application development teams can think of security as just another aspect of software quality. Applying the same concepts and disciplines as designing for reliability and performance will work here too. You might suffer a short term investment but for a long-term gain. Chances are your customers are demanding secure products, so why not embrace this non-negotiable business driver as a necessary activity in your product development lifecycle? Run the numbers for yourself and determine your total cost and the risks associated with introducing new activities into your development process.

So what is a good balance between overwhelming your team with an entirely new security process and introducing new activities to help you minimize the impact of security defects in your software? An effective development process assumes that code will be attacked and embeds some kind of assessment at each phase as a health check. Here are some keys to getting traction in your organization:

- ✚ Keep the changes lightweight: enhanced security practices need balance between impact to development time as well as actual investments made
- ✚ Promote awareness by educating individuals that play key roles in your development process. They need to understand the importance of security and what the fundamental goals of the increased security efforts are to your business
- ✚ Remember that security is about mitigating risks at a given cost. Get senior management buy-in for security programs by demonstrating how the new activities mitigating business risk.
- ✚ Build in checkpoints at each phase to ensure that key activities are performed, i.e., define minimum criteria that must be met before the application “passes” that phase.

There is no short-cut to building more secure software, but there is a way to view the cost and risk of insecure software that helps you justify additional time and dollar investment. Specifically, look at security activities that track and identify risks – activities like code reviews, threat modeling, and secure requirements analysis. These present ways to view tangible and “predictable” consequences of insecure software and identify “hidden costs” that often escape notice.

Security is very difficult (and too risky) to bolt in after the fact (see misconception number one in this series: over relying on network defenses.) Many companies maintain the philosophy that to have secure applications they need a separate security team or they need to redefine their entire SDLC; neither is true. Start light, but before you start at all – run the risk numbers and determine for yourself how important secure software is to your organization.

## 5. Falling into the “Recency” Trap

This is a psychological problem more than anything. People tend to react to the most recent scare. We’ve heard about the Ernst and Young employee leaving a laptop on a bus. ING had sensitive data on a laptop that was stolen. Events like these cause an interesting psychological phenomenon I refer to as “The Recency Trap” – changing your habits based on perceived risks caused by recent events of which you’ve become aware. The mistake organizations make here is overreacting to perceived threats, which in turn allows for more real or serious threats to go unattended.

This common information security mistake is the antithesis of the false sense of security one gets when overlying on network defenses (Mistake #1) or believing the hype of tools (Mistake #2).

### *A Sense of Falling*

When high-profile cases like E&Y, ING, and the VA hit the public conscience, organizations tend to react by making adjustments in their security spend. The incidents of lost laptops or data tapes led to organizations rolling out new policies to encrypt all data on laptops and in databases. Not to say that laptop and database encryption is a bad policy, per se, but you have to understand that it only mitigates the risk of physical theft of the computer or database. Further, it still does not protect the organization from risks such as hacking, malicious code, bots, worms, trojans, insider crime, un-authorized workers who steal proper authorization, criminal third parties, etc. And this all assumes that encryption was implemented correctly and the organization provided ample training for their teams, which almost never happens. The driver for the new policy was a perceived new or heightened threat. This is psychology driving security spend and it is a dangerous trap.

The same thing happens in the non-IT world, too: in 1967 Sweden changed from driving on the left side of the road to driving on the right. What happened? In the 12 months following, auto fatalities dropped by 35%. Was the right side of the road safer? Was there a major advancement in automobile safety in Sweden in 1967? No. There was simply a change in rules and as a result people felt more at risk due to the recent “incident.” The sad part is that twelve months later, auto fatalities were exactly where they were pre-1967. People “forgot” they were at risk and adjusted behavior once again.

### *Rising to the Occassion*

When asked to qualify why you’re seeking security budget dollars, ask yourself whether the amount of spending is absolutely critical or if you’re just doing it to cover your butt. CISO’s and CIO’s are frequently in the position of having to justify security spend. If the drivers are eerily similar to ones mentioned above, consider the reasons carefully before asking for those dollars.

Perhaps you’re even asked to implement a certain security solution because your board recently became aware of a security incident in a similar type of organization and they’re fearful the same might happen at your organization. While you’re asking yourself questions, ask this one, too: “If I alone had the ability to decide where and how to spend security budget, and I could only choose one place, where would I spend it?” Often the answer is different than where your board might choose - and forcing the issue can make you a voice of reason in a world of security FUD (fear, uncertainty, and doubt.)

### *PCI in Question*

The Recency Trap has even bitten compliance regulations like PCI. In my opinion, the PCI DSS (Payment Card Industry Data Security Standard) is one of the best efforts put forth to date to secure a company’s information systems. It isn’t perfect, but it is highly prescriptive and provides very clear guidelines on

what to do (and not do) to provide a more secure computing environment when handling credit card data. With the recent data security breaches at TJX and Stop & Shop, PCI has come under fire. After all, if both of these companies were PCI compliant, how could they have been breached so easily? There are two things to remember here:

1. Compliance does NOT equal security
2. The Stop & Shop breach was a carefully-planned and well-orchestrated attack, specifically targeted at their point-of-sale card swiping kiosks. It was anything BUT an easy attack.

The fact that the PCI standards council is coming under fire because of these attacks is a shame; and it shows again that The Recency Trap is a powerful psychological gripper. When people feel threatened, they want to point fingers; it's only human to want to identify someone or something to blame for your feelings of insecurity. Unfortunately, this often drives us to make knee-jerk reactions and poorly-considered decision. PCI is not to blame here; in one incident it was very savvy criminals and in the other it was an organization that hid bad policies and mismanaged their applications and data.

### ***Threat Modeling: The Recency Trap's Arch Enemy***

Falling into The Recency Trap is dangerous, but we're all prone to the psychology – we're human. One tactic to beating off the pitfalls of psychology is threat modeling. Threat modeling is an important activity for risk management and I describe it in more detail below. It helps you identify which assets (or liabilities as I prefer) are susceptible to particular threats. In the absence of threat modeling, you are more at risk for psychological over-reaction.

When you develop a threat model it also becomes a sustainable asset. If a new vulnerability or a new threat is detected, you can reuse your threat model to determine whether or not you are at state of heightened risk, decreased risk, or neutral. A basic tenet of threat modeling is that threats are realized through scenarios that can be exploited via vulnerabilities if not mitigated with appropriate countermeasures. But you'll never know which countermeasures are best for you if you don't analyze the system first!

## **Avoiding Mistakes: Five Good Practices**

As valuable and occasionally humorous as the above mistakes can be, the real pay off comes when you understand what proactive steps to take to prevent your organization from making these same mistakes.

Below I provide five practical tips to get you on your way. Of course, every organization's mitigating controls are highly contextual, so adopting all five may not be right for you – but if you allow these to serve as a beacon, you will be much more informed about information security and better equipped to make decisions on time and resource investment.

This short five-step plan will help you to integrate security into your information management and application lifecycle and each is a short-term investment for a long-term gain – the best of both worlds as security is fast becoming a non-negotiable business requirement that your customers are demanding.

### **1. Make a Self-Assessment**

This is quick and inexpensive. It means going through a check list to see if you have incorporated application and information security into your risk management framework and determine whether you

have integrated security into each phase of the software development lifecycle. It is a very simple meeting with your VP of application development to have him or her list the different phases of their specific software development process. Then ask how they handle security at each phase and determine whether or not the outputs of those activities are usable in your risk management process. If the outputs aren't useful, perhaps you should be measuring something different. In most cases, the answers you get will be something like, "Well, we've just started thinking about how to integrate security into our application development, so we don't really have anything tangible for you at this time." That's OK because that would be an ideal time to discuss your needs with that team. Bridging the gap between application development and risk management is a highly valuable activity and it can be jump-started by this simple self-assessment. It's a simple checklist that will give you a quick gap analysis as to where you stand on the information and application security maturity model (see figure 1 below).

Threat modeling is also an important and valuable step in a self-assessment. It is a more mature and sophisticated approach than the checklist mentioned in the previous paragraph, but the payoffs are substantially greater. Threat modeling, at the business level and the application level, is part of a risk analysis and risk management that allows you to identify where the biggest threats are to your business. This is the Sun Tzu approach of "To know your Enemy, you must become your Enemy." The basic idea is to define a set of attacks or negative scenarios and assess the probability, potential harm, priority, and business impact of each threat. From this point, you can define measures that can minimize or mitigate the threats, which in turn help you make investment decisions. This can be done at \_any stage, e.g., design thru deployment and yields more valuable results the earlier it is applied. You may need help on your first couple of threat modeling exercises, but there are plenty of good information security consultancies that can provide this. You can also find additional threat modeling resources from organizations like NIST, Microsoft, and Security Innovation.

When you develop a threat model it becomes a tangible, persistent asset for your organization as well. If a new vulnerability or a threat is detected, you can reuse your threat model to determine whether or not you are at a risk increase, decrease or static. The threat model can help you avoid falling into the recency trap and will tell you whether or not a newly-identified threat is already mitigated in your system. It is a tool to help you make informed security decisions.

## **2. Believe the Application Security Hype**

This is an unfortunately necessary action as there is a lot of hype and fear out there that vendors and media are spreading unnecessarily. However, the application security hype is very real and we have seen it from recent and past headlines – the Lexus-Nexus breach, the recent problems at TJX, and even the incidents at T-Mobile and CardSystems were all information security incidents caused by application security holes. So how do we filter through the chaff to determine what is real and what isn't? The most effective way to do this is to focus on the application layer. The network and systems layers represent less than 30% of all security vulnerabilities (according to Gartner Group); this number is less than 10% according to NIST. Also consider that network security is much more mature than application security and the investments you have already made here are probably orders of magnitude higher than those you've made in application security. Don't ignore network security, but try this practical tip: make a list of the investments and compensating controls you have in place on the network (e.g., anti-virus, IPS, firewalls, etc.) and their associated costs (initial/purchase cost is fine). Now do the same for your information and application security investments and compare them. If the application security investments aren't at least 2-3 times that of network security you have an imbalance in the number of vulnerabilities you're mitigating.

With that first filter applied, now consider where and when it is most costly to address application security. IDC and IBM conducted a study about two years ago that mapped the cost of fixing a problem

through the software development lifecycle. The results were roughly exponential with respect to time and phase. Said another way, if a defect found at the design phase costs 1X to repair, the same defect costs 6.5X to fix if found during the coding phase, 15X if it were found in the pre-deployment testing phase, and 100X if it's found by your customers in the field after it is deployed. And keep in mind that this only accounts for the time and effort it takes to fix the problem (internal costs). It doesn't even factor in things like reputation loss, cost of patching and deploying, and other losses that usually come along with security defects -- things like loss of market share and stock price.

### 3. Ask Tough Questions

Tough questions are great because they force you to think. The challenge is usually in determining what questions are the tough ones. I provide a short list here to get you started. You will see the pattern fairly quickly and can expand on them for your own environment. These are questions that are useful to ask your vendors before making a software purchase. You should also ask the same questions of yourself as you build and maintain applications for your business to use. Finally, use these questions to improve your service-level agreements with outsourced partners (esp. software development partners). You are probably making a purchase or partner contract to automate or transfer some business function -- shouldn't you also consider how to mitigate or transfer your risk when you're doing this?

- What is your vulnerability response process?
- What is your patch release strategy?
- What methods do you use to inform customers of vulnerabilities?
- What guidance do you provide for secure deployment/maintenance of your product?
- What security training does your development team receive?
- Do you patch all versions of your applications at the same time?
- What are the terms and period of your security support agreement?
- Do you practice security reviews at each phase of your software development lifecycle (requirements, design, coding, testing, and deployment)?
- Do you employ independent 3<sup>rd</sup>-parties to conduct security assessments on your products?

### 4. Create an Internal "Red Team" of Ethical Hackers

This is a term borrowed from the military. The concept here is that you dedicate a small team -- usually 1, 2, or 3 people -- to act as attackers. This can be a permanent role if you can afford the resources, or you can take some nasty-minded testers and make this part of their job at various phases in the development process. Their job is to attack your application systems and your networks as if they were malicious (and I don't recommend constraining them to act as outsiders.) The insider threat is often overlooked and you can learn a lot from creating attack scenarios from an inside user perspective. If you don't have the resources or skill set to create Red Teams, there are many third-party consulting shops that can do this for you. Start with your most critical applications and work your way down the risk rank stack.

If your organization doesn't have the luxury or skill to create a Red Team, you should look to outsource this function to an independent 3<sup>rd</sup> party application security assessment firm. There are many good options available to you and often you can find multiple solutions available in one vendor, for example, a company that specializes application security assessments might also be able to provide technical training for your teams.

By the way, you also need to be certain that any third party assessments company you use is capable and credible. So ask those same tough questions of them, focusing on things like: methodologies used,

credentials, engineers they are going to use to test your application, how/if they depend on the use of automated tools, etc.

## 5. Educate your Teams

I cannot understate the value and importance of this practice. Education is the first step towards awareness – and as you will see in the chart from Gartner below, you still have a long way to go after you have become “aware!” The challenge most organizations face here is two-fold: How to best educate their teams, who might be geographically disbursed and of different skill set; and, which team(s) to invest in for security training.

Deciding which team to train (or in what order) is a highly contextual decision that needs to be made based on your specific organization. However, having helped several companies successfully roll out security awareness programs recently, I have observed a few critical success factors that I will share here:

### Management buy-in

An important step toward success in any security awareness or training program is management buy-in. Security awareness will likely lead to behavior and policy changes at your organization; for that to happen effectively and efficiently, management must be on board. Even better – make them part of the change by ensuring that your program has elements that appeal to management.

### Ensure policies can be enforced

Write clear, understandable, current, and measurable policies. Naturally, the policies need to reflect the corporate, threat and regulatory environment. Awareness and training programs should address the importance of adhering to policies, as well as the potential financial and reputation impact to the organization from security events.

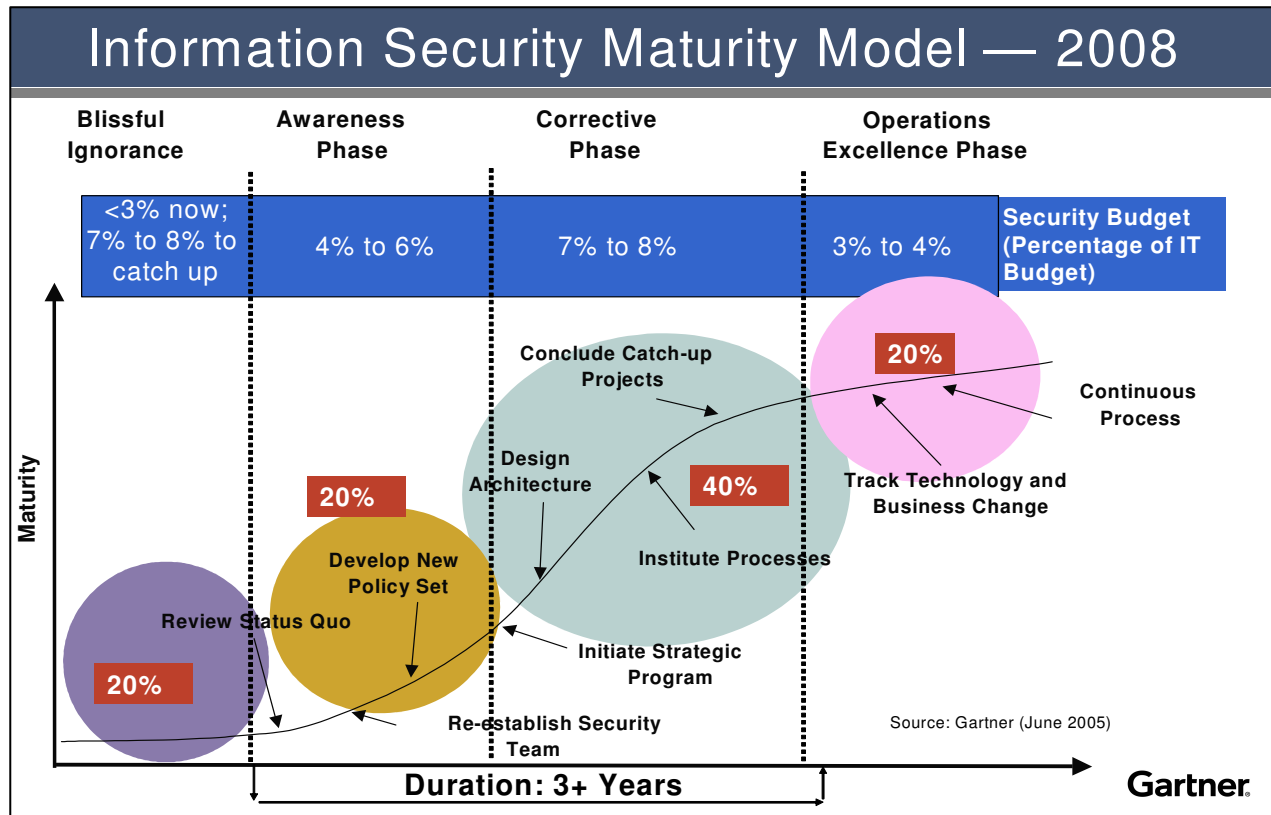
### Measure and Report

Use both qualitative and quantitative metrics to obtain feedback, measure and benchmark the effectiveness of your security awareness and training program. Most importantly, communicate these metrics and results (good or bad) to your management team for their input, support, and insight.

If at all possible don't limit education to only security awareness, but also provide technical security training for your engineers, auditors and others. This training is more difficult to find, but you can locate some excellent security specialists that provide training in scalable formats, e.g., eLearning, for both management and technical staff.

## An Analyst's View of Security Investment

Below, I provide a chart created by Gartner Group. It describes what they call the Information Security Maturity Model, or ISMM. The chart shows the progress organizations make as they mature in their information security awareness. It tracks the percentage of organizations' IT budgets allocated to security and shows how it balloons and then contracts as companies move through awareness toward operational excellence.



I find it interesting that 80% of organizations are still in either the “blissful ignorance,” “awareness,” or “corrective” phase. I suspect that number is substantially higher if this were tracked for only application security. The message I take from this is: GET AWARE. And the best way to start? Well, if you’ve made it this far in the article, you are well on your way!

### About the Author

Ed Adams is the President and CEO of Security Innovation, the independent authority on application security risk assessment, risk mitigation and education. He is a seasoned software executive with successful business experience in various-sized organizations that serve the IT security and quality assurance industries.

As CEO, Mr. Adams applies his technical and business skills, as well as his pervasive industry experience in the Application Quality space, to direct world-renowned application security experts and deliver world-class professional services to many of the most recognizable companies in the world including Microsoft, IBM, Visa, Fedex, ING, Symantec, SAP and HP.

Mr. Adams is the founder and business owner of the Application Security Industry Consortium, Inc. (AppSIC), an association of industry technologists and leaders establishing and defining cross-industry application security guidance and metrics. He is on the board of the National Association of Information Security Groups (NAISG).

No stranger to the podium, Mr. Adams has presented to thousands at numerous seminars, software industry conferences, and private companies. He has contributed written and oral commentary for business and technology media outlets such as New England Cable News, *CSO Magazine*, *SC Magazine*, *CIO Update*, *Investors Business Daily*, *Optimize* and *CFO Magazine*.

Mr. Adams is in the process of writing a book titled "Information Security Management: Survival Guide", which will be published by Wiley & Sons and is due out in August of 2007. He also has maintains a blog with *CSO Magazine* and is a columnist for *CIO Update*.

### About Security Innovation

Security Innovation is the authority on application security and a leading provider of risk analysis, risk mitigation and education services to mid-size and Fortune 500 companies. Global technology vendors and enterprise IT organizations rely on our services to identify security risks in their software systems and development processes and facilitate the changes needed to mitigate them. The company is headquarter in Wilmington, MA and has offices in Seattle, WA and Amsterdam, the Netherlands.