Microsoft®
Security Development Lifecycle

Microsoft SDL: Return-on-Investment

September 15, 20099/15/2009 4:19:00 PM

For the latest information, please see http://www.microsoft.com/sdl.

Trustworthy Computing

iSEC PARTNERS

ABSTRACT

Times of economic contraction often put security programs under pressure as targets of budget cuts. The return on investment (ROI) for security activities is not always clear because:

- the benefits produced by security programs, assets and investments are not always apparent despite their existence
- measuring the results and progress of security activities is difficult; some of the best results might be when impacts to business are completely mitigated and nothing bad occurs

Taking a systematic, structured approach to secure software development can produce measureable benefits well beyond the traditional risk reduction calculations, and organizations can achieve meaningful results with a smaller investment than many decision-makers realize. A well-managed, structured software security program is a good investment at any time.

This paper will help managers:

- Understand and communicate the benefits of a structured approach to software security.
- Develop and use metrics for ROI to guide process improvement.
- Get meaningful results from a new program or optimize existing efforts on a limited budget.

Investing efficiently in security requires that you understand the organization's goals, along with how to measure progress toward them and how to evaluate the return on your investment. With a structured approach to security activities, organizations can achieve immediate benefits in several key areas:

1. Fixing vulnerabilities early leads to lower overall development and maintenance resource expenditures.
2. A well-functioning security program can produce competitive advantages by reducing business process violations, bringing predictability to security costs, and increasing agility in changing compliance regimes.
3. Good metrics that are customized to an organization's specific goals enable organizations to measure the effects of activities on overall security and support the decisions an organization must make to balance resource expenditures and to improve over time.

INTRODUCTION

Software is critical to the process of doing business for almost all organizations. Although recognition of the importance of secure systems is growing, software security must still compete for a place in an increasingly tight enterprise budget. IT decision-makers must clearly understand why the investment involved in software security is justified. In this white paper, we show that a well-optimized security program can reduce the overall cost of developing an application and the business processes that it enables.

This paper examines the case for software security by looking at the return on investment (ROI) that is generated by a well-managed security program. Organizations frequently underrate the ability of security to contribute to the competitiveness of an enterprise, and the tools available to understand these contributions are not widely applied. Managers can use ROI and security metrics to build a financial case for software security programs and to demonstrate how they can accomplish meaningful improvements within a reasonable budget.

An effective software security program does not necessarily require huge budgets or massive structural changes, but it does require a structured approach and a culture shift within your development organization. There are affordable ways an organization can maximize the effectiveness of their security engineering efforts over time. Managing vulnerabilities at the correct stage of the software lifecycle can even save on total development time.

SECURITY: WHY YOU SHOULD CARE

Security can help make your organization more efficient and generate ROI in multiple ways—security is not purely a sunk cost. A more secure system reduces incident response costs, and helps prevent violations of business processes. A well-managed software security program reduces the cost and risk of securing systems and regulatory compliance. Finally, effective security measures integrated into the software development lifecycle can reduce the cost of vulnerabilities (both security and otherwise) by improving overall quality.

Security incidents are costly and unpredictable, and in today's environment, any application is likely to be targeted. Responding to an incident carries direct costs, including:

- Fixing the exploited vulnerability.
- Determining which systems were compromised.
- Rebuilding compromised systems.
- Communicating with customers.

The cost of investigating and remedying violations of business process may be very large. In 2008, confidentiality breaches of customer information cost affected organizations an average of $202 per customer record and $6.6 million per incident.[1] Although these numbers include an estimate of indirect costs, the effect on an organization's reputation, loss of customer confidence, and customer churn can be detrimental, especially if your industry requires public reporting of major incidents. Other incident types, such as loss of critical intellectual property or disruption of business, can cost much more and may be impossible to estimate or budget for.

[1] According to the United States 2008 Annual Study, "Cost of a Data Breach," by the Ponemon Institute and PGP Corporation.

Even if incidents are considered somewhat unlikely, the costs of investing in security can be better than the alternative. Security investments are predictable and can be budgeted for. In these times of tight budgets and economic stress, the business risk from the unexpected cost of an incident is higher than ever, and computer crime is only expected to increase.

Organizations with more mature security programs generally have much less work to do to comply with new regulatory regimes. Compliance with these regulatory requirements tends to be very costly for businesses that are not prepared, and the general trend of the past few years toward increasing regulation can be expected to continue. Therefore, developing a security program when it suits the organization, beyond the immediate benefits, can be a strategic advantage as unprepared competitors struggle to catch up.

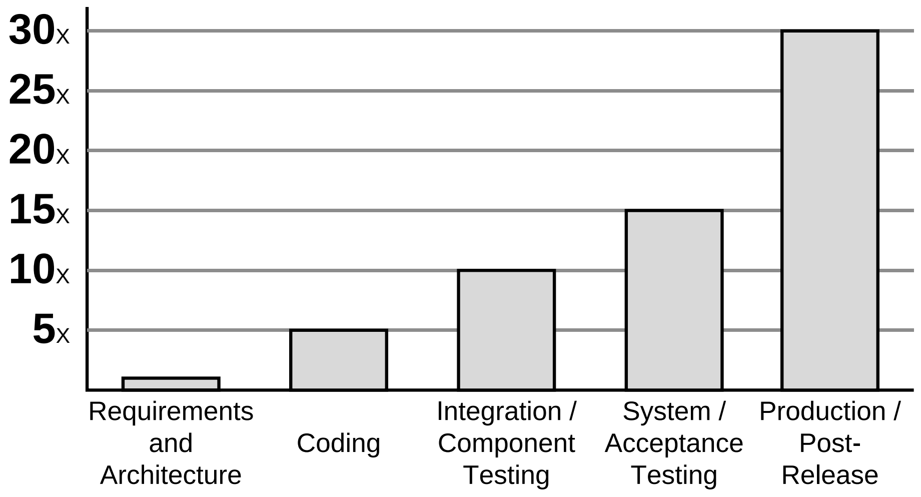
A structured approach to secure development is fundamentally intended to produce higher quality, more reliable systems. Vulnerabilities that are found and eliminated before production are much less costly to fix, and systems that are more secure require fewer patches and cost less to operate. Downtime and business process errors are reduced, regardless of whether the system is under attack.

WHY A STRUCTURED APPROACH PRODUCES A SUPERIOR ROI

Often organizations that are beginning secure software development programs are tempted to add on security testing at the end of the development process. It is easy to plan the budget and schedule for a security review of a largely finished application; findings are clearly spelled out, and it does not require changing the development process or distributing work and responsibility across multiple reporting chains. The results from a penetration test can bring attention to the need for a software security program. However, this approach generates a lower ROI than a structured approach that performs security efforts up front, early, and throughout the development process.

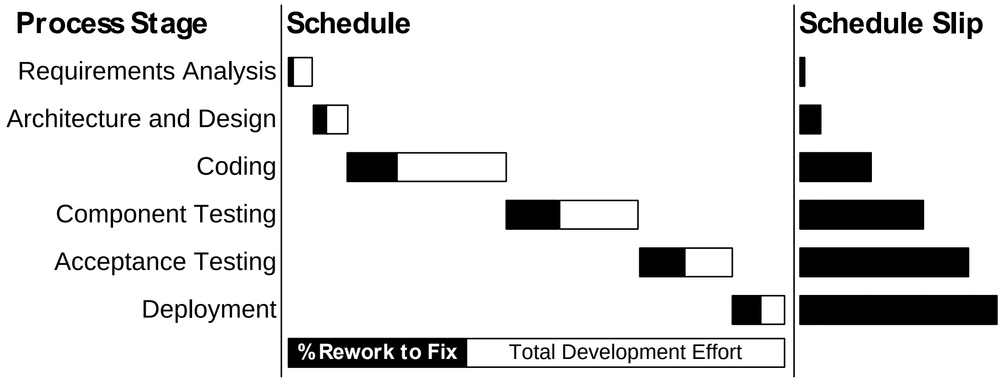
One major reason that a security program that spans the development process is ideal is that show-stopping security vulnerabilities discovered very late in the software lifecycle are much more time-consuming and expensive to fix. A study by the National Institute for Standards and Technology[2] (NIST) estimated that the cost of fixing a vulnerability that was not discovered until acceptance testing might be up to 30 times greater than if the vulnerability was fixed during the design phase. Reacting only to vulnerability reports from late-cycle security reviews and incident reports from production systems consumes more resources, introduces more schedule risk, and increases pressure to ignore an issue in order to hit a ship date.

[2] According to the 2002 NIST Report, *The Economic Impacts of Inadequate Infrastructure for Software Testing*.



Relative Cost to Fix, Based on Time of Detection

Consider a security deficiency that exists at the requirements level. Depending on the level of security involvement, it may be caught at a number of different stages. Although the schedule is likely to slip in all cases, and additional resources will be required to fix it, the stage that it is caught at significantly effects the expenditure. Because every product and organization is different, we do not assign numbers here; the relative proportions of effort are intended to illustrate the cascading effect. The chart that follows shows the chain of dependencies with the approximate proportion of rework at each stage that is required to fix a discovered deficiency in the system's security requirements.



If a security review identifies the issue at the requirements stage, some rework is required to determine the correct system behavior and to revise the requirements. If the issue is not caught until the design phase, it is still necessary to go back to the requirements phase to understand how to change the system behavior, and now the system architecture also has to be changed. As soon as the actual code is being written, it becomes much more difficult and time-consuming, especially if the vulnerability is expected to have architectural implications. Even small changes to system architecture may require rewriting large portions of code.

Incorporating proactive security efforts at each phase adds a cost to the schedule, but that cost is predictable and is frequently quite small. Security issues that flow down through several levels in a process increase in complexity rapidly, like any large system change, and the effect of a single architectural vulnerability may be larger than the total cost of several preventative security practices. Vulnerabilities that are discovered after deployment may have extremely large relative costs, from active exploitation or just from the opportunity cost of having to shut down live operations or features until the vulnerability can be fixed. The exact cost/benefit tradeoff is different for every organization; your metrics should provide the information you need to understand this situation.

No silver bullets

An attacker needs to find only one vulnerability in a system to exploit it, while the defender needs to find and fix as many as possible until he or she reaches the required level of system security assurance. The Microsoft Security Intelligence Report (SIR) shows that the vast majority of vulnerabilities being exploited are in applications, as opposed to the browser or OS.[3] Application developers for large and small software projects all need to invest in secure development practices. Without a structured process, a development team cannot understand how close it is to completeness within an application. Similarly, without a structured process, a management team cannot track completeness across its family of applications or understand the level of assurance that it needs. Regardless of the quality of the individual security activities, it is the structured process that makes those processes reliable and functional. Additionally, the structured process provides the framework for operating that allows you to measure and improve the quality of individual activities.

The inequality between attackers and defenders has another important consequence: There can be no silver bullets. Many security products on the market claim to be able to “fix” your software security problem. Some of these products are extremely useful to security teams, when they are used with a clear understanding of what they can and cannot do. Used correctly, these products can act as force multipliers for the available resources, but you cannot rely on those products alone or use them effectively without an ongoing commitment of time and expertise. All such tools have blind spots, such as when an attack exploits business logic deficiencies in a way that cannot be distinguished from legitimate user behavior.

These tools have a place in a complete structured approach. It is appropriate to add them at different times, depending on the needs of an organization and the available budget, but it is important to realize that no product can ever substitute for secure software development. Security must include the people, the experts, and the larger development organization, a culture shift towards security, tools where they are useful, a security process that makes everything work together, and a set of metrics that allows for improvement and deeper understanding of process.

HOW TO MANAGE SHRINKING BUDGETS

Security is often considered a nonessential activity when budgets must be cut. Unfortunately, as long as insecure applications are in use, the overall risk that they pose to business processes does not change. In

[3] http://www.microsoft.com/sir

fact, as applications age and new categories of attack are discovered, applications that are not updated are at a higher risk of being exploited. Maintaining a culture of security is even more important. If an organization takes a stop-and-go approach to security, development groups may believe that security is not a priority, so they will not put effort into the work—even when economic conditions change.

As we have shown, eliminating security activities as a way to cut budgets can actually reduce operational efficiency and lead to increased overall costs. In tough economic times, the risks of fraud increase as the ability and resources to respond to a catastrophic security event decrease. Understanding how to use ROI calculations to show explicitly why security investment is worthwhile is key to defending your system against counterproductive cutbacks.

HOW TO TELL IF A SECURITY INVESTMENT IS WORTHWHILE

Every organization has different goals for security activities. To know if a security investment is achieving its purpose, you must clearly define the goals of that investment. Reducing the number of possible business process violations is a common high-level goal that organizations can easily quantify by using threat modeling and related activities. An investment in training might have as its goal a lower total number of security vulnerabilities, while an investment in threat modeling might have a goal of increasing the ratio of vulnerabilities discovered during design compared to those discovered in later phases. After you identify goals for your activities, you can determine how you will measure and quantify their benefits to ROI.

Although security metrics are still an emerging area, with a solid understanding of the benefits you hope to realize, you should be able to devise good metrics to make your budget case and to serve as ongoing decision-support tools. Every organization has different goals, and different metrics will be appropriate to track those goals and support the decisions the organization needs to make.

Metrics need not be expensive to create or track. The simplest possible metric that accurately captures the information you need is usually the best one. Complex metrics can require exponentially more work and are frequently unreliable because of incorrect embedded assumptions.

A full understanding of security ROI begins by understanding what we want to avoid. For example, each potential business process violation can be associated with a specific, calculated cost of recovery. You can look at each security action and see how it can reduce the number of potential business process violations and how much that action costs. With these two costs and a comparison factor, you can determine an ROI. In an ideal world, the comparison factor would be a reliable probability, but meaningful probabilities of compromise do not exist. An organization with high-value targets in a high-threat environment (like Microsoft) might operate under the assumption that *all* vulnerabilities will eventually be discovered and exploited, but most organizations have to decide where to place this bar.

Ideally, all security metrics would be concrete and tied to specific business processes, but this kind of business-driven analysis is not always possible. Luckily, simpler metrics are often sufficient. The number of vulnerabilities that are found and fixed is a good starting metric for the success of security activities that are intended to find vulnerabilities. When comparing applications, it is important to evaluate this metric in the context of the security maturity of the application and the level of effort related to finding vulnerabilities. To make this metric more concrete, you could relate vulnerabilities to an estimated average cost or worst case cost of recovering from the exploitation of an issue, which can be useful for comparing systems.

For security activity in the development process, a useful metric could be the number of violations of security policy, such as using known unsafe APIs. Measuring the savings from finding or avoiding issues early in the development process and comparing those savings to the cost of fixing issues in production can be very effective. In organizations where the teams which bear the costs of security activities are not directly accountable for their results, you can build credibility for security activities by justifying the cost savings in terms of these teams' own key performance indicators, such as developer days saved or schedule slip avoided.

Metrics that are appropriate for each stage of a project can be used to create an incentive to build in security throughout the project. These metrics should be tracked in both the short term and the long term. In the short term, they provide information about specific projects; in the long term, they give information about organizational maturity.

Different kinds of security activities pay off in different ways. Post-development security activity shows up as vulnerabilities are caught—a direct measure. Pre-development activity, including training, shows up as a reduction of vulnerabilities that are introduced into the system. Your metrics should help you understand ambiguities (for example, "Do we have great training or poor verification?"). It may be difficult to determine what a metric is showing without additional information. Try comparing applications where teams did and did not receive security training. Or try looking at how changes in the types of vulnerability that are found in an application correspond to the training a team has received.

Although metrics are used primarily for short-term security assessment and long-term process improvement, they can also be important tools to justify future resources and to sell security activities to management. Sometimes it may be more important that the metrics that are selected match up with the expectations of management than that they are as simple as possible. Balancing both organizational purposes is important.

In the end, pick the metrics that matter for your business and that support the decisions you need to make. Identify your goals and targets, quantify the costs of leaving them unaddressed or the positive returns for meeting them, and measure the effects your security investment is producing in each area. Repeating these processes over time allows you to understand both the security profile of specific applications and the security maturity of your entire organization.

HOW TO MAXIMIZE YOUR INVESTMENT

For a security effort to provide the maximum ROI, it must be conducted efficiently. If you already perform some security activities, but you want to allocate your resources more efficiently, you can:

- Increase the organizational maturity of security processes to best take advantage of work done.
- Concentrate your efforts at the beginning, where they will have the most effect.
- Make sure you get incident response right, because that is where you may be spending money—whether you want to or not.
- Make sure you get what you pay for when performing activities and selecting tools.

Increase organizational maturity

Organizations which have mature security processes produce more secure software with less involvement from a centralized security team. They also waste fewer resources on repeated process changes and lack of familiarity with processes, and they produce a more standardized and well-understood product. A good process improvement plan is important for long-term success in increasing organizational maturity. The Microsoft Security Development Lifecycle (SDL) Optimization Model[4] is an excellent resource to draw on for identifying effective security practices. It can also help you evaluate your current state and get prescriptive guidance on how to begin or improve the maturity of your security program in a structured way. The Building Security In Maturity Model[5] and the Software Assurance Maturity Model[6] provide alternate ways to measure your organization's security maturity.

Even if you are not ready to adopt a full maturity improvement model, you can begin with some key activities that provide the foundations of a mature security program: training, organizational standards and policy, and governance of the security program itself.

Training is a critical part of organizational maturity. Personnel involved in security activities must understand the work they are being asked to do. Training should cover not just security fundamentals (recognizing vulnerabilities and preventing them from being introduced) but, for the broadest audience, should also cover the security standards, procedures, and responsibilities specific to your organization. The Microsoft SDL – Developer Starter Kit provides a compilation of baseline developer security training materials to help organizations get started.

Standards and policy can start simple—complex and overly comprehensive standards are not always efficient or more effective than lightweight standards. Your standards should provide the team with technical information about how secure applications are to be developed—including banned APIs, mandated best practices, and mitigations for different types of vulnerabilities. Policies speak to higher-level issues about the development process and how systems that implement different categories of business processes handle different classifications of data or how systems that interact with different types of environments must be secured. Effective standards provide a reference point for architects and the security team to work from when they evaluate applications and create security specifications. Effective standards also provide developers and testers with a resource for acting on those documents. Likewise, policy can help ensure shared understanding of how processes are intended to work and can empower a development team to speak up when they see problems looming.

Good governance is key to a mature security program. Processes for issue-tracking, documentation, and assignment of responsibility must be created in conjunction with the teams that they will affect, and they must be integrated with policy. The details about the roles that are needed for interactions between development and security—and about which of those groups will perform specific duties—vary significantly between organizations. The maturity models provide more information on the topic. However, it is critical to the long-term success of security efforts that you ensure that everyone is comfortable with the resulting agreement and has a shared understanding of it. Common elements of the agreement include assigning responsibility for completing tasks, allocating resources between development and security, tracking vulnerabilities and applications, and dividing authority among groups.

[4] Available on MSDN here: http://msdn.microsoft.com/en-us/security/dd221356.aspx.

[5] Available here: http://www.bsi-mm.com/.

[6] Available here: http://www.opensamm.org/.

Communication is also critical to the security process. Overly complex documents and detailed processes may be inefficient in some development environments, but all organizations should have a basic set of templates, a standard communication process, and a defined set of escalation procedures during development and in the event of an incident. Without good governance and communication, the work you do will disappear: Potential issues will not be resolved, incidents will be mishandled, and vulnerabilities identified in the testing process will slip into production.

The same metrics covered earlier in this document are also critical to improving organizational maturity. Without a way to measure your enterprise's security activities and their effects, it is impossible to understand whether the steps you are taking are having any effect, whether they are paying for themselves, and what you should do next. Collect data, apply metrics, and act on that information to see long-term improvement.

Concentrate at the beginning

Because the cost of fixing vulnerabilities throughout the product development lifecycle goes up, all projects should involve security from the very start, in the form of a requirements review. This review allows the security team to identify high-risk applications quickly and to find vulnerabilities that are inherent in project requirements. Requirements-level vulnerabilities happen more often than one would expect and can be far more costly to address since they can impact the actual core functionality in a manner that undermines the original business goal. For example, if a system design permits two users to collude and trigger actions neither is allowed to perform, this creates a vulnerability even if the system is implemented perfectly. A good requirements process supports the discovery of security issues before development. Discovering these issues before development is critical, because requirements vulnerabilities are frequently difficult to fix later. A well-defined workflow for creating requirements is necessary to ensure that issues are discovered early, and organizations can benefit from the more formal process involved. Data collection for metrics should begin as requirements are being created to ensure that software is properly tracked and handled across its entire life span.

A high-level formal architectural security review should be part of the design process, whenever possible, frequently through threat modeling. The high-level review is necessary to find architectural vulnerabilities in a coordinated manner, similar to the function of requirements review. Doing a review for architectural vulnerabilities during design means that more of these vulnerabilities can be mitigated when it is least expensive—before code is written. Having a good threat model for an application makes future security review activities simpler because the security implications of the architecture are already understood. A high-level architectural review requires a significant effort by experienced personnel, and, although a it finds issues more efficiently, organizations should balance high-level and low-level reviews to avoid missing implementation vulnerabilities. If you have enough resources to provide full beginning-to-end review coverage for high-risk applications, the high-level review should be expanded to more applications because it can help locate unexpected inter-system interactions and similar vulnerabilities that can change expected risk profiles.

A security specification should be created from the results of the requirements and high-level security analysis to ensure that this knowledge is captured and passed into development. Some issues can be eliminated at the design and requirements level, and the security specification can provide guidance on most others. As the organization matures, creation of this document will become cheaper and faster. This stage provides another useful touch point for measuring security progress.

When an organization begins a training program, architects should be trained at the beginning of the process—before testers. In the same way that fixing vulnerabilities early is more efficient, architects, who are leaders in both the technical and cultural standards of the development organization, are best positioned to make an immediate impact with additional knowledge. Giving a few architects enough training to make a measurable difference is also more realistic and affordable than giving even basic training to a much larger group that starts with a lower skill level. Developer training should focus first on ensuring that they understand how to correctly implement features noted in security specifications, standards, and policy. Later, as resources allow, training can expand to a larger secure development curriculum. Track training to determine its effects on vulnerability levels that are found in the field.

Get incident response right

Incident response is an exception to the general rule of starting at the beginning. Spending a large amount of money to build an incident response organization before you develop security requirements is a mistake, but incidents happen. If you have prepared for an incident with at least a basic plan, including defining points of contact and escalation, response, and communication procedures, incident response will go more smoothly and will be less disruptive—which, in turn, saves money. Eventually, an incident will happen, and you will spend the money either way. If you do some work on incident response early on, you can spend the money when it works for you, instead of when you have no choice, and the costs from the incident may be much lower.

If you do have an incident, treat it as the source of valuable information that it is. Root cause analysis should be a part of all incident response plans, and the knowledge gained should be integrated into future software security efforts and used to validate and calibrate metrics.

Get what you pay for

One of the most common ways that organizations waste money on security is by not getting what they pay for, whether they are buying services and tools or performing activities internally. While organizational maturation and high-level, up-front review activities almost always pay off, code-level security efforts are more complicated, from an ROI perspective.

Organizations often put unnecessary constraints on teams that are looking for vulnerabilities, hampering their efficiency and wasting valuable resources. Give your internal or external test teams everything they need to do their job as quickly and completely as possible, including source code, a running test system, internal documentation, and developer access. Remember that defenders must find all vulnerabilities, while attackers only need to find one. Furthermore, not all attackers will be outsiders. The idea that "black box" testing alone is superior because it replicates attacker perspective is a fallacy. Use the one structural advantage you have—inside knowledge—to your fullest benefit.

Automated tools can also fall short of expectations. They must be evaluated with a realistic understanding of their capabilities and in light of their actual total cost, including personnel, training, and tuning. The initial cost can be high, but the ongoing effort of configuring, operating, and managing the output of these tools is higher, and they are just one part of a security process, not a complete solution. Take advantage of the available free tools to estimate these costs and your organization's capability to use a more complex and expensive tool effectively.

In the context of an ROI-focused effort, your metrics should provide you with information about the relative utility of the different measures you use. Pay attention to these metrics. If they are good, they will show you how to optimize your processes and get what you pay for.

Keep measuring

The most fundamental step in moving your organization toward a more efficient process is to keep measuring. To be useful in an enterprise context, security must be measurable and organized. If you continue to measure your ROI and improve the process, you will increase the efficiency of your organization. At the same time as you tune your process with metrics, review the metrics to ensure they still make sense for the new process. Coarser ROI metrics will help you allocate resources among different efforts, and finer measurements will help maximize the effectiveness of individual activities.

Process optimization example

As an example, we will look at an enterprise that has been previously performing black box penetration tests of all of their projects (about 20 a year) but does not have a structured approach to software security. It has put two development teams (one-third of the organization) through a basic secure coding course with the intent to continue working through the rest of the organization, but the outcomes have not been tracked, and personnel have moved between teams. The enterprise is not currently under any regulatory constraints that interact with computer security, but management is expressing concern that this may change. There is also pressure to bring down the cost of existing security efforts.

Following the recommendations mentioned earlier, they make a number of changes in their current practices. For each change, they define a set of metrics to determine both the initial and ongoing value of the change. By careful allocation of budget and time, they manage to reduce the overall cost of security efforts across the organization while measurably improving results—achieving their stated goals. It is important to note that these may not be the goals (nor the constraints) of all organizations. For instance, it may be the case that budget is simply fixed, and the resource being optimized for is time. Alternately, both time and budget may be flexible, with the goal being the fewest possible unplanned schedule disruptions for security-related issues.

Enterprise-scope security activities:

Existing activity	Cost	New activity	Cost	Metrics used
Secure coding training for 100 developers/year	$200k	Extensive training for 20 architects/year	$100k	Requirements and architecture-influenced vulnerability delta
		Standards and policy training for 300 developers/year	$40k	Standards compliance delta
		Annual source code scanner license	$50k	Verified vulnerabilities filed Full-review vulnerability delta for similar applications
		Standards and policy creation and updating	$10k	Vulnerability delta for covered issue types

Existing activity	Cost	New activity	Cost	Metrics used
		Incident response planning	$5k	Incident cost delta versus industry, response time
		Communication and process planning	$5k	Cost delta for equivalent security activities
		Enterprise-scope metrics tracking	$10k	N/A
Previous total	**$200k**	**New total**	**$220k**	

Project-scope security activities:

Existing activity	Cost	New activity	Cost	Metrics used
Black box penetration test for 20 projects/year	$1,200k	Full access security review for 10 projects/year	$800k	Severity-normalized vulnerabilities found per person-week
		Requirements review for 20 projects/year	$80k	Requirements-level security vulnerabilities, relative project risk, business-process violation cost estimate
		Security analysis/threat modeling for 20 projects/year	$120k	Architecture-level security vulnerabilities, relative project risk
		Security specification creation for 20 projects/year	$40k	Vulnerability delta for covered issue, types versus similar applications
		Security vulnerability tracking	$20k	N/A
		Project metrics tracking	$20k	N/A
Previous total	**$1,200k**	**New total**	**$1,080k**	

Based on this, we can look at the ROI for a specific item. While some of these cases are more involved, requirements reviews provide a good starting point.

The Widget Tracking System is intended to handle order fulfillment for an internal supply chain; outside suppliers interact directly with the system. The requirements analysis finds ten different business process violations of interest, and the estimated single violation occurrence cost for each one ranges from $10k to $150k. Of these, five can be mitigated entirely through checks and balances put into place at the requirements level, three of these at little or no cost, and the remaining two at a recurring cost of approximately $100 per transaction. The low-cost options are an easy choice, but determining if the higher cost options are worthwhile requires understanding how frequently transactions are likely to be affected.

Unfortunately, there is no way to calculate a simple probability here. Any specific calculations will require more knowledge about the set of potential attackers than is available. What we do know, however, is the amount of attacker effort required to realize the attacks, and we know the relative difficulty. Both of these are valid metrics, and in the absence of any information on whether an attack will occur, the only responsible action as a defender is to assume that it will and to scale responses according to the difficulty of the attack.

In this case, of the two attacks with recurring mitigation costs, one is trivial, easily justifying the increased transaction expense of protecting against it. The other attack requires a significant investment on the part of the attackers, and the cost of developing an exploit is likely more than it is worth; the mitigation is not justified on the part of the defender and should instead be documented as an assumed risk. This covers the ROI analysis embedded in the requirements review. Once the review is done, we can look at the total cost of the potential business process violations and compare that with the total cost of the activities (both in terms of analysis and increased development time) to determine the level of benefit received from the activity. There must be a scaling factor here; again, we do not have a realistic source for probability as a scaling factor. Instead, organizations should decide on the scaling factor that they wish to use, since it reflects the organization's risk tolerance and desired level of assurance. In the end, this is a policy decision, not a metrics one; the metrics can only make clear exactly what the decision is.

STARTING FROM ZERO WITH A SMALL BUDGET

Starting a new software security effort is easier and less resource-intensive than it looks initially. You can start from scratch with relatively few resources and still get results, if you have management support. As the work pays for itself, justifying resources will be easier. Some management support is important for efficient progress. Constraining the initial resource requirements and providing a clear plan for how you will measure ROI can help get this support. Some of the existing process improvement models described earlier in this paper may be useful here, as well, but these six activities are a good starting point:

- Plan carefully before you work to structure your process.
- Start at the beginning to leverage structural efficiencies.
- Select pilot projects carefully, paying attention to risk management.
- Build appropriate standards and policy.
- Create an incident response plan.
- Get expert help when appropriate.

To work efficiently, you must balance activities that will result in the highest return immediately and activities that will make the overall effort most efficient, even if they don't return an immediate payoff.

Start with a plan

Before you start any security activities, build a plan. The plan should cover not only basic execution but should also become the basis of a structured security process. Equally important in this plan are the metrics you use to determine ROI. You should know how to measure the effectiveness of actions before you take them.

Take advantage of the flexibility of not having an existing security process. Quality gates before products ship are a necessary component for verifying system security and enforcing policy, but gatekeeper-only processes lead to excessive last-minute work, make fixing vulnerabilities expensive, and do not encourage collaboration between security and development. Instead, create a process that involves security from the beginning of development processes.

The cultural shift towards security is critical for success. Avoid stopping and starting or randomizing the way you roll out security activities. In evaluating which security activities to undertake, there is frequently a desire to try a few things out in small doses. In other cases, there will be a push to "start doing security," which may result in first one and then another process or technique being rolled out across an organization without a larger plan. This scattered approach can slow or stop the cultural shift.

Start at the beginning

Start a new security process at the beginning, the same way you improve an existing process. Talk to the development teams you will work with to ensure that you understand their development methodology and the project lifecycles. Find out how they measure things internally and what data they already collect. See how the existing metrics and data can most efficiently fit into the metrics you plan to use.

Start your training at the beginning. Turn your architects into security experts and champions, and let that expertise inform requirements and design activities. Track training internally to measure its efficacy.

Start your security process while requirements are being written with security-focused requirements analysis. Use a high-level security analysis process, like a threat model, to evaluate the architecture of systems. Write a security specification based on results of the requirements analysis and the threat model. In each case, track your data. Some of these artifacts will be useful later in the testing process or in the next version of the application, and some of them may also provide useful data for metrics.

Manage risk and select your pilot project

When you start a new security program, pick the first projects carefully. Even if you intend to ramp up the program to cover all projects within the organization, it needs to begin with a pilot project. Consider a project with security risk as a pilot; a project with a security risk increases the likelihood that the security issues with demonstrable business impact will be found, helping to justify future efforts, optimize skills, and ensuring good resource utilization right away. However, no matter how well-factored the new process is (and even if you can train the entire team up front), new processes always have quirks that take time to work out. To increase the probability of a successful pilot, avoid choosing an expensive, very high-profile, tightly scheduled, or otherwise resource-constrained project.

To make this kind of decision for a pilot project, you need to be prepared to make decisions about the relative risk of projects. Metrics are key here. Developing the metrics up front to evaluate future projects is important. Using these metrics from the start can improve their acceptance and let you see trends sooner, although sometimes pilot projects must be exceptions. For example, set a bar that requires any project with a business process violation cost estimate above a specific value to receive a full review, and all applications that will be hosted on a public network must receive at least a basic review.

Add standards and policy

Standards and policy are useful for new security efforts, but delivering a complex set of standards right out of the box is a bad idea. Standards must evolve with the culture. As always, make sure you have a measurement plan for new policies and standards. The plan can be quite simple. For example, if you specify a standard mitigation for a class of vulnerabilities, track how the number of vulnerabilities in this class changes in new development. Similarly, track compliance, because it provides useful information to determine whether a standard is ineffective or is just being ignored—each of which suggests a different response.

Take a balanced approach with new standards, specifying process-level issues, requirements-level issues, and code-level issues. Specifying only process-level and code-level issues makes it more likely that the organizational culture will develop with the understanding that security is something that happens at the end of the process instead of a holistic process that encompasses the entire lifecycle.

Incident response

Incident response is not a good initial focus for a large set of resources, but as we mentioned in the section on optimizing resource allocation earlier in this paper, you should pay some attention to incident response early in your structured security planning.

Get expert help

When you start from scratch, it is important to have experienced support. Some existing programs, like the SDL Optimization Model, are designed to help organizations new to structured security (or to security in general) improve the maturity of their processes. In addition, it can often be worthwhile to bring in outside consultants with significant experience to help you avoid process pitfalls. A good consultant can work with you to understand the needs of your organization, help you develop the processes that will work for you, and even help deliver customized standards and policies. Furthermore, outside consultants can be a good way to cross-check your progress in terms of security maturity, both at a process level and at an individual system level. They can calibrate the metrics you use and provide spot checks on specific systems by using application reviews, even if you are staffing most of your security activities internally.

New program example

A small development organization is considering a new security effort but is unsure about the costs and their capabilities. They have a team of approximately 50 developers working on about five projects a year. Management is very interested in security, in theory, but they are resource-strapped, in practice. Being able to fulfill multiple requirements with the same activities is a priority.

Following the recommendations described earlier, they build a fairly small and very front-loaded process to take the maximum possible advantage of structural efficiencies in the absence of a legacy process. They start with a single pilot project for a full security review, and they bring in outside experts to help. In addition, they start to do high-level review activities on two more projects. As data comes in from these initial activities, they expand to a larger set of activities. In addition to looking at the security benefits, metrics are chosen to examine the overall reliability and quality of the systems to help justify the new security expenditure in light of overall efficiency.

Enterprise-scope security activities:

Activity	Cost	Metrics used
Extensive training for five architects/year	$25k	Requirements and architecture-influenced vulnerability delta
Standards and policy training for 50 developers/year	$10k	Standards compliance delta
Standards and policy creation and updating	$5k	Vulnerability delta for covered issue types
Incident response planning	$3k	Incident cost delta versus industry, response time

Activity	Cost	Metrics used
Communication and process planning	$3k	Cost delta for equivalent security activities, non-security development maturity
Enterprise-scope metrics tracking	$2k	N/A
Total	**$48k**	

Initial project-scope security activities:

Activity	Cost	Metrics used
External full-access security review for one project	$80k	Severity-normalized vulnerabilities found per person-week
Requirements review for three projects	$16k	Requirements-level security vulnerabilities, business-process violation cost estimate, non-security project quality improvements
Security analysis/threat modeling for three projects	$15k	Architecture-level security vulnerabilities, non-security project quality improvements
Security specification creation for three projects	$8k	Vulnerability delta for covered issue, types versus similar applications
Security Vulnerability Tracking	$2k	N/A
Project metrics tracking	$2k	N/A
Total	**$123k**	

Eventual Project-Scope Security Activities:

Activity	Cost	Metrics used
Internal full access security review for two projects/year	$120k	Severity-normalized vulnerabilities found per person-week
Requirements review for five projects/year	$20k	Requirements-level Security vulnerabilities, business-process violation cost estimate, non-security project quality improvements
Security analysis/threat modeling for five projects/year	$30k	Architecture-level security vulnerabilities, non-security project quality improvements
Security specification creation for five projects/year	$10k	Vulnerability delta for covered issue, types versus similar applications

Activity	Cost	Metrics used
Security vulnerability tracking	$2k	N/A
Project metrics tracking	$2k	N/A
Total	**$184k**	

Half the battle is knowing how large the investment is and knowing the metrics we are tracking. Now we need to look at how these activities improve our resource utilization.

Cost estimates for vulnerabilities classes

These cost estimates are based on the cost of fixing a vulnerability based on the phase in which it is discovered. Although using real data is optimal, any metrics on vulnerability costs can be used to calculate this for pre-release vulnerabilities; they do not necessarily have to be for security vulnerabilities. Additional costs are added for high- and medium-severity vulnerabilities discovered in the deployment phase to account for actual incident losses or downtime to patch.

Severity	Requirement / Design	Development	Integration test	Acceptance test	Deployment
High	$1k	$3k	$7k	$15k	$80k
Medium	$1k	$3k	$7k	$15k	$40k
Low	$1k	$3k	$7k	$15k	$30k

Number of vulnerabilities discovered or prevented during initial activities:

Activity	Vulnerabilities discovered	Savings versus discovery in production
External full access security review for one project	2 high	($80k x 2) - ($15k x 2) +
	3 medium	($40k x 3) - ($15k x 3) +
	10 low	($30k x 10) - ($15k x10) =
		Subtotal: $355k
Requirements review for three projects	1 high	($80k x 1) - ($1k x 1) = $79k
Security analysis/Threat modeling for three projects	1 medium	($40k x 1) - ($1k x 1) = $39k
		Total savings: $437k
		Cost of work: $123k
Total	**3 high, 4 medium, 10 low**	**$473k - $123k = $350k ROI**

This clearly shows the return received on our investment, since the cost to fix these issues before production is much lower than the combined cost of fixing them earlier in the process and the cost of the security activities throughout the lifecycle.

Choosing between alternatives

Here, we will look at another example of ROI, this time in balancing two activities. To justify security expenditures, we must use metrics with the denomination upper management responds to. In this case, we will assume that metric is the vulnerability count. In evaluating whether working with existing, in-house security staff to perform code reviews is more or less efficient than bringing in outside consultants, we can start by comparing the costs for each. Assuming that internal resources are cheaper, but less experienced, we can then perform pilot projects (ideally, multiple instances of each, but this may not be possible) and look at the results. We will want to scale the number of vulnerabilities found by the amount of time spent and some estimate of the severity, using a standard rating system. Even with very limited data, we have three possible outcomes: Either the two will be roughly cost-comparable, per vulnerability, or one or the other will show a significant difference. A single project's results may be anomalous, but if future activities do not bear out these results, having a statistical baseline will allow us to see the difference quickly and to communicate this difference to management in terms they will listen to. While this does not provide a dollar-for-dollar ROI, it does provide a more important ROI scaled in terms of the benefit the organization cares about.

CONCLUSIONS

Improving the security of a system makes it more reliable and less expensive to operate in multiple ways. While software security efforts require some resource commitment, you can achieve a significant ROI, often with a small initial expenditure. Careful use of metrics allows you to measure the effects of your investment, and those same metrics allow long-term improvement of security ROI and overall effectiveness.

A structured approach to security makes the process more predictable, dramatically improves its efficiency, and allows the security team to deploy their resources in a heavily leveraged, top-down manner. A combination of high-level analysis, low-level review, metrics-based risk management, and tools will provide an optimal, measurable ROI.

When organizations structure their development processes to involve security practices as early as possible, the cost to fix many vulnerabilities decreases dramatically. While tools should be part of the equation and can provide a force multiplier, no product can substitute for secure software development. An effective, structured approach to software security must include people—both experts and the larger development organization—a cultural shift toward security, tools where useful, the security processes to tie activities together, and metrics that allow for understanding and improvement.

Security metrics are at the heart of all these elements. They provide a means to understand the effects of disparate security activities, to balance resource expenditure, and to improve security processes and organizational maturity over time. Well-managed structured security is always a good investment.