Software Engineering Institute | Carnegie Mellon University

The Role of Computer Security Incident Response Teams in the Software Development Life Cycle

Robin Ruefle

January 2007

ABSTRACT: This article describes one type of organizational entity that can be involved in the incident management process, a Computer Security Incident Response Team (CSIRT), and discusses what input such a team can provide to the software development process and what role it can play in the SDLC. CSIRTs in organizations performing software development and in related customer organizations may have valuable information to contribute to the life cycle. They may also be able to learn valuable information from developers concerning the criticality, operation, and architecture of software and system components that will help them identify, diagnose, and resolve computer security incidents in a more timely manner.

INTRODUCTION

Incident management activities, while not specifically called out in the software development life cycle (SDLC), are an important part of the maintenance, operations, and sustainment of any software or hardware product. Decisions made during the SDLC, from user interface design to providing facilities for patch management, can significantly change the likelihood of incidents and the success of any response to them.

Knowledge gained from detecting and responding to computer security incidents provides insight into real risks and threats to the integrity, confidentiality, and availability of software and hardware products. This information can be used at the beginning of the SDLC to help define better security requirements in products and provide a better understanding of the threat environment within which these products must operate. Knowledge gained from containing and mitigating computer security risks and threats can also help identify auditing and recovery requirements for systems and software. Such requirements can include building in alerting capabilities when files and components that should not be changing are modified, establishing policy and configuration setting capabilities to identify and control specific software and hardware components that should not be changed during normal operations, or providing functionality for logging unau-

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

thorized changes or malicious attacks in a manner that would preserve evidence in a forensically sound manner.

Although computer security incident management may seem to come at the end of the SDLC, the more knowledge from such activities are applied throughout the design, development and implementation of operating, application, enterprise systems and networks the more effective both the SDLC process and the incident management process will be.

In the long run, feedback from incident management activities can be used to produce systems that are easier to manage, have reduced operational risk, are less impacted by cyber attacks, and have improved networked systems security and survivability.

Knowledge of the intricacies of how specific software and hardware components function and interface with each other is critical to understanding how those systems are at risk, how they can be exploited, and how incidents can be successfully mitigated. This type of information is usually known by the systems' developers, administrators, and owners. Incident management staff, especially in a large software-diverse organization, may not have this knowledge. Being able to rely on the expertise of developers for assistance in analyzing not only the risks but the best resolution strategies for systems can decrease the time it takes to contain and recover from malicious activity.

How successfully information is collected and shared between product developers and incident management staff will depend not only on the group responsible for incident management activities but also on the structured relationships between that group and the system developers.

Best practices in CSIRT development and implementation call for the CSIRT to identify staff and departments throughout the enterprise with whom they must coordinate and work. Although this list includes many throughout the IT, security, and management structure, it very rarely if ever includes system developers. This article will build a case for why such an interface is important and will explore some methods for encouraging such interaction.

Before proceeding, some basic introductory information on what a CSIRT is and what it does may be required. For details, see the BSI article Defining Computer Security Incident Response Teams.

CSIRTS AND THE SOFTWARE DEVELOPMENT LIFE CYCLE

Computer security incidents can often be the first place where symptoms of wider, ongoing problems are noticed. Such information—if properly captured and disseminated—can help to identify root causes of vulnerabilities, recurring security problems and incidents, and mitigation and resolution strategies related to such incidents and vulnerabilities. Historically, this information has not been passed to developers or introduced into the SDLC. Incident response personnel or CSIRT staff have not generally had an established interface with developers. This is beginning to change as more people in both fields (security and software engineering) have begun to understand the benefits of such information sharing.

There are many points in the SDLC where information exchange between these two communities can provide useful information for designing better systems as well as handling incidents on those systems.

Because CSIRT staff know what type of information they need in order to understand and resolve incidents, they will have their own set of requirements for how detection, response, and remediation processes should be built into or at least supported by software systems and applications. These types of requirements need to be incorporated early in the design phase. Such requirements might relate to what type of event logging is required at an application or host level, what type of a notification is required for alerting staff to unauthorized system changes, or what type of functionality would allow for rapid containment. A good example of incident response functional requirements might be designing software that is forensically friendly. This means designing software that has been engineered to capture information and evidence in a forensically sound manner, providing a record of where information came from and how it was collected in a way that cannot be repudiated. When collecting incident data or evidence, developers could work with incident handlers to understand how the data is to be analyzed, used, and archived as part of the response process. There may be specific information that can be collected and structured for automatic input into a tracking system or repository or collected in a way to increase the completeness and confidence in the information.

Many of the topic and content areas within the BSI Web site describe components of the SDLC where knowledge between developers and CSIRT staff can be exchanged. Many topics provide best practices and guidance, outlining both requirements for better software and systems and ways to improve the development process. Reviewing some of these requirements and methods identifies multiple areas where information on software and hardware vulnerabilities and malicious attacks and events are used to identify potential risks and threats to the security and quality of products. Understanding the cause, history, and mitiga-

tion of these risks and threats can provide information that can be used to design better software requirements, model threat environments, test and evaluate software security and recovery components, and measure software survivability.

A CSIRT, by virtue of its mission and function, is a repository of incident and vulnerability information affecting its parent organization as well as its constituency. This information can be used to provide real life risk and threat information. It can also be used to provide guidance for reviewing and testing the security components and requirements of developed or acquired hardware and software.

The following sections support this assertion by presenting various topics from the BSI collection and highlighting where they call out the need for information that could be provided by a CSIRT. The following sections also discuss some potential interfaces where CSIRT information and bi-directional interaction can be introduced into the SDLC.

Software Assurance

The draft document Security in the Software Life Cycle states that “Software assurance … includes the disciplines of software reliability…, software safety, and software security.” Software security is defined in the same document as “the ability of software to resist, tolerate, and recover from events that intentionally threaten its dependability.” The document goes on to say that

The objective of software security is to design, implement, configure, and support software systems in ways that enable them to

1. "Continue operation correctly in the presence of most attacks by either restricting the exploitation of faults or other weakness in the software by the attacker, or tolerating the errors and failures that result from such exploits;
2. "Isolate, contain, and limit the damage resulting from any failures caused by attack-triggered faults that the software was unable to resist or tolerate, and recover as quickly as possible from those failures" [Goertzel 2006].

Because of the nature of CSIRT activities, information is continually collected on how various exploits and vulnerabilities work, what new and emerging trends and threats are developing, and what new mitigation tactics and strategies work best to isolate, contain, limit, or eradicate any damage. CSIRTs often perform these analysis and response functions or coordinate with other groups in the organization that perform these functions.

Knowledge from such activities can be used to develop abuse and business cases that can be used in the development of system requirements as well as operational exercises that can be used to test the security of system components. This information can also be used to understand the root cause of vulnerabilities so that preventive measures and configurations can be built into software and hardware. Information on how vulnerabilities are exploited can provide guidance to software developers on how to eliminate those vulnerabilities by building defenses into systems, software, and hardware. Knowing what intruders are trying to obtain or compromise identifies what needs to be protected and gives software developers information on what components need to crash or recover gracefully without exposure to exploitation.

As active participants in the handling of computer security incidents and vulnerabilities, CSIRT staff understand the type of tools required to perform such work efficiently and effectively. CSIRTs can provide information to system developers about what software functions need to be built into products to support incident detection, analysis, and mitigation activities.

Security Requirements Engineering

Of course it makes sense that CSIRTs and software developers would work together during the requirements elicitation phase of the SDLC, specifically in defining security requirements. The BSI site describes requirements elicitation as "best practices for security requirements engineering, including processes that are specific to eliciting, specifying, analyzing, and validating security requirements." In Security Requirements Engineering, Mead talks about various methods that can be used to help define security requirements specific to particular applications and then test that those requirements are being met [Mead 2006b]. With their understanding of attackers, motives, targets, and techniques, CSIRTs should be involved in any security requirements elicitation activity to provide different views and probabilities for what type of attacks might be realistically executed. In Requirements Elicitation Introduction, Mead reviews numerous elicitation methods ranging from controlled requirements expression (CORE) to misuse cases to Joint Application Development (JAD). CSIRTs would be valid participants in any of these methods as they can help identify security problems based on historical incident data and trends as well as predictions of future intruder activity.

Attack Patterns

Another strategy for understanding software risks and corresponding mitigation strategies is the area of attack patterns. Attack patterns, as described by Barnum and Sethi, provide insight into methods used to exploit, compromise, and basi-

cally “break” software. According to the authors, attack patterns are “an abstraction mechanism for describing how a type of observed attack is executed [Barnum 2006a].” The attack is described from the perspective of the intruder or attacker

The BSI content area related to attack patterns lists common fields of information that are captured to describe the details of an attack [Attack Patterns 2006]:

- Pattern name and classification
- Attack prerequisites
- Description
- Targeted vulnerabilities or weaknesses
- Method of attack
- Attacker goal
- Attacker skill level required
- Resources required
- Blocking solutions
- References

Such information can be very useful in the development of software security requirements. In Introduction to Attack Patterns, Barnum and Sethi explain how understanding attack patterns can provide software developers with insight into the real environment in which software and hardware must exist. Software developers can then use this information to develop software requirements and resulting products that are not only more resistant to such attacks but able to recover more quickly when attacked.

The incentive behind using attack patterns is that software developers must think like attackers to anticipate threats and thereby effectively secure their software. Due to the absence of information about software security in many curricula and the traditional shroud of secrecy surrounding exploits, software developers are often ill-informed in the field of software security and especially software exploitation. The concept of attack patterns can be used to teach the software development community how software is exploited in reality and to implement proper ways to avoid the attacks [Barnum 2006a].

CSIRT staff and their incident and vulnerability repositories are valuable valid sources of information about current and new attack patterns and trends. CSIRT staff can also serve as subject matter experts to provide an in-depth perspective on how these attacks and corresponding mitigation strategies work and how they can be translated into software development requirements. Through their day-to-

day work, CSIRT staff can help distill the patterns that are evident across various attacks. Organizing joint sessions to brainstorm new attack patterns or review existing patterns could be one way that managers stimulate interaction between software developers and incident management staff. The output of this interaction can provide input into software requirements for products to be able to defend against such attacks.

It is not enough to just understand the specifics of one attack. Looking at the higher level problem or the pattern across the attacks is what will help developers build more secure and resilient software. In Attack Pattern Usage, Barnum and Sethi state that

Attack patterns can be an invaluable resource for helping to identify both positive and negative security requirements…Many vulnerabilities result from vague specifications and requirements…Requirements should specifically address these ambiguities to avoid opening up multiple security holes [Barnum 2006b].

Attack patterns can provide information on common security flaws and problems. These problems can be addressed in various parts of the SDLC. Initially, they can be used to identify those security requirements that need to be met through the development and design phases. They can then be used to help generate sample attacks that can be used to test that the software security requirements have been met and that the software reacts to the attacks in the desired manner.

In general, attack patterns allow the requirements gatherer to ask "what if" questions to make the requirements more specific. If an attack pattern states "Condition X can be leveraged by an attacker to cause Y," a valid question may be "What should the application do if it encounters condition X?" [Barnum 2006b].

During the testing phase, the same attack should be executed or simulated and then the question becomes "Does the application react to condition X according to the identified security requirements?" This should be followed by the question "Does the reaction prevent the attack from succeeding (or at least mitigate its impact)"? CSIRT and other incident management staff can help design and execute these tests and corroborate that the reaction is the desired one.

Other questions might include "Are there any other conditions that would cause Y to happen?" or "Can condition X be leveraged by an attacker to cause a different result?" Again, CSIRT experience in the areas of incident and vulnerability handling and mitigation, along with collected historical attack information, can be used to help create and understand attack patterns and corresponding mitigation strategies. This information, as previously said, can then be used to develop

software requirements related to preventing, deterring, or mitigating such attacks.

Threat Modeling

Threat modeling, in a general sense, addresses who would want to attack you, how they might do it, and what resources they might use to do it. CSIRTs should be actively participating in such threat modeling to help software developers understand who might want to attack their applications, what these attackers are looking for (i.e., what's valuable to them), and what type of techniques they might use to perform the attacks. CSIRT staff can also use their experience and knowledge of the intruder community to help identify how realistic different attack scenarios will be based on their understanding of the likely threat. CSIRT staff can contribute case studies for threat modeling and also explain new trends, emerging attack techniques, and intruder behavior and motivations.

Threat modeling has also been defined as "a structured approach for identifying, evaluating, and mitigating risks to system security" [Goertzel 2006]. This is another area where input from CSIRTs' real-life experiences, expertise, and research can be used to help determine new and emerging threats and risks. Goertzel and colleagues recommend building risk analysis throughout the development process. Risk assessments can be used to help verify that identified risks and threats are being adequately addressed or handled through either safe and secure software failure modes, secure configurations, implemented auditing and alerting mechanisms or planned incident response procedures. CSIRT staff, although not usually trained in risk assessment methodologies, can provide input about how their organization's critical systems and data may be at risk. They may participate on the assessment team or be interviewed by the assessment team as subject matter experts.

Architectural Risk Analysis

The Architectural Risk Analysis section of the BSI Web site discusses the process for conducting architectural risk assessments. The Architectural Risk Analysis document defines this process as "a risk management process that identifies flaws in software architecture and determines risks to business information assets that result from those flaws" [Hope 2005].

The document outlines the process and lists the key steps as

- Asset identification
- Architectural risk analysis (threats and vulnerabilities)
- Risk mitigation
- Risk management and measurement

Here again, CSIRT case studies, situational awareness, and expertise can provide information into the SDLC. By virtue of their expertise and day-to-day operational tasks, CSIRTs collect, track, record, and analyze real-life information on threats and vulnerabilities that may impact their parent organization or constituency's enterprise. As previously discussed, they can also provide input about attack patterns and identified historical risks. All of this information is useful in performing architectural risk analysis.

Architectural Risk Analysis is described in the IEEE article "Bridging the Gap between Software Development and Information Security" by Gary McGraw and Kenneth van Wyk, as one of a series of touchpoints where coordination and data sharing between software developers, risk analysts, and CSIRT staff (along with other information security experts) can provide real-life information on the types of risks and threats that software developers must take into account when designing and implementing software and systems [McGraw 2005].

Touchpoints in the SDLC

McGraw and van Wyk go on to describe other touchpoints that, like Architectural Risk Analysis, illustrate how the expertise and lessons learned from information security staff and specialists (e.g., CSIRTs) can enhance secure software development efforts. These touchpoints, which are presented as best practices that can be implemented in the SDLC in an effort to allow for collaboration and coordination between the normally isolated areas of information security and software development, include

- "developing abuse cases that can be used to help refine requirements and build business cases
- performing business risk analysis
- implementing test planning such as security functionality and risk-driven testing
- performing code review
- performing penetration testing
- deploying and operating applications in a secure …environment" [McGraw 2005]

Abuse Cases

It's easy to see how CSIRTs can provide real examples for developing abuse cases. They know the real abuse that goes on within their own environment and study abuses that happen to others. They share information on attacks with other CSIRTs, read articles on such cases, and attend conferences to learn about this

type of information. They work to understand the technical details of attacks and vulnerability exploits and can use this knowledge to develop abuse scenarios.

Business Risk Analysis

Working with auditors and risk analysts, infrastructure groups, and software developers, CSIRTs—whether internal to the software development organization or coming from the organization's customers—can provide input into the types of risks and threats that may impact business operations. By itself, the CSIRT is not generally the group that calculates the business impact, but it is important that they understand the business impact so that they can prioritize response actions. The CSIRT though, through their experiences, will be able to explain how various attacks and compromises can occur, providing the true details of the risk. Auditors, along with financial and risk analysts working with the groups that maintain and support enterprise software and systems, can use this information to determine what impacts such malicious activity will have on the infrastructure and thereby the business.

CSIRTs within customer organizations can provide more in-depth information about their own environment and the risk and threats they face as well as the general remedial assistance they require in handling computer security incidents. This group may be a good reference for software developers to understand customers' day-to-day incident handling requirements. Engaging customer CSIRTs through focus groups, as part of a needs analysis or software design processes, may alert developers to common problems that can be remediated in the initial software development activity.

Software Testing

Although the CSIRT is not the group that will normally perform any software testing, they can provide test scenarios based on their real-life experiences or research. Real-life scenarios will provide a good method of testing how well software stands up to and handles current risks as well as what type of security functions have been configured and implemented. Just because they are not generally part of the testing group does not mean that the CSIRT shouldn't be part of the testing group. This could be another approach to ensuring interaction between developers, testers, and incident management specialists. CSIRT staff could focus any testing they might do on looking for common secure coding errors and vulnerabilities.

Code Reviews

McGraw and van Wyk state that it is generally not the information security staff (CSIRTs, in our case) that reviews code to ensure that security bugs and weak-

nesses have been removed or mitigated [McGraw 2005]. However, some members of CSIRTs may actually have the expertise to do this type of review. There is a small cadre of CSIRT staff and security experts who are well versed in certain malicious code and artifact analysis techniques—such as surface analysis (including source code review) and reverse engineering techniques (disassembly and decompilation of binary code)—who have the skills to do a security code review. However, many teams may not always have the resources and time to allow such proactive activities to occur. However, it may be possible, though, to take some general lessons learned from CSIRT artifact analysis work and synthesize these into best practices that can then be incorporated into secure programming methods, techniques, and training. Such information may also be used to implement security quality testing as part of a code review.

There are a few books on secure coding that have been released over the past few years that present these types of best practices. These include

- Graff, Mark G. & van Wyk, Kenneth R. Secure Coding Principles and Practices. Cambridge, MA: O'Reilly Media, 2003.
- McGraw, Gary. Software Security: Building Security In. Boston, MA: Addison-Wesley, 2006.
- Seacord, Robert. Secure Coding in C and C++. Boston, MA: Addison-Wesley Professional, 2005

Penetration Testing

Depending on the services provided, a CSIRT may perform penetration testing as part of its operational activities (either by request or on a periodic basis with management approval). If the CSIRT is performing this function, it is important for any weaknesses and vulnerabilities discovered in systems to be relayed back to developers so that they can fix current problems and also incorporate these lessons into any future software development activities.

If the CSIRT does not perform penetration testing, they again can provide real-life exploit scripts and malicious code that might be incorporated into a penetration testing toolkit that can be used by other groups in the enterprise that are responsible for performing this activity. Use of such malicious code, though, should always be done cautiously.

CSIRTs often keep a database of such exploit and malicious code in their incident tracking system or in its own secure repository. Such code can be analyzed for comparative purposes or to identify patterns and trends in exploitation across

protocols, applications, or operating systems. It can also be used to help build anti-virus and intrusion detection signatures.

Another significant area where penetration testing can play a part in the SDLC is discussed by van Wyk in his BSI document Adapting Penetration Testing for Software Development Purposes. He states that penetration testing is usually not incorporated early enough into the SDLC, being done after a product is deployed rather than as part of the testing phase. He recommends involving security and incident management staff earlier to perform penetration testing to help determine significant security problems before products are placed into production. This will allow for remediation of risks that, if left until deployment, could open software and hardware components to significant malicious impact [van Wyk 2006].

He also discusses the benefits of black box and white box penetration testing approaches. “In a black box test, the tester starts with no knowledge whatsoever of the target system, whereas in a white box test, the tester gets details about the system and can assess it from a knowledgeable insider’s point of view" [van Wyk 2006]. White box testing is one activity where CSIRT or other incident management staff and software developers could collaborate. This is one way to develop an interface or communications channel between these diverse groups. The software developers have the detailed view of how the components operate, and the CSIRT or incident management staff can apply this knowledge during penetration testing exercises.

Deployment and Operations

McGraw and van Wyk argue that deployment and operations should be viewed as part of the SDLC [McGraw 2005]. If accepting this argument, then this provides another touchpoint where CSIRT staff can have input into the SDLC by providing information that can be used during deployment and operations activities to prevent incidents and protect critical assets. Specifically CSIRTs can provide best practice guidance for implementing secure default network and system configurations to prevent incidents. They, by virtue of their position, will also be the key players in instituting any type of incident reporting guidelines and response plans for handling computer security incidents that do occur.

CSIRTs collect and analyze data to determine the cause of computer security incident as well as the correct strategy to mitigate the resulting danger. Information resulting from this analysis can be fed back into the software design, deployment, and operations phases to help prevent similar incidents in the future.

The deployment and operations phases of the SDLC should include activities to detect, analyze, and respond to computer security incidents. Performing such activities successfully is critical to the sustainment of any software and hardware product. Software design should also then include requirements for supporting these activities. Since CSIRT or incident management staff generally perform these functions, they can be a good source of information for building such software requirements. Involving CSIRTs early in the SDLC, through joint design meetings or requirements setting interviews, can help determine what software products require to be easily remediable. For example, most incident handlers will agree that software logging features are not adequate and the currently available analysis and archival tools are not optimized or easy to use. Searching and correlating information can be difficult and retaining incident information in a way that can be reviewed to help understand current problems and needed remediation leaves much to be desired. CSIRTs can provide information to software developers concerning what information should be automatically collected as well as how that information should be stored and structured to maximize and support analysis, trending, and response activities.

Obtaining customer or product CSIRT input into the design of user interfaces and deployment features can provide a reality check that software is indeed easily remediable. An example of this could be building software in a modularized fashion so that patching can occur in an optimized way. Another can be ensuring that patches themselves are not the same size as the original program.

CSIRTs can provide recommendations for the best security configurations for applications and even provide some pre-incident planning and preparation recommendations that can help developers support the CSIRT mission. For example, in prioritizing incidents, it is important for CSIRT staff to understand not only the criticality of a service or server but how that component fits into the total enterprise architecture, what trusted relationships it has with other components, what data is contained on the system, and what data is shared (including how it is shared and with whom). The CSIRT can obtain some of this information from the system developers and administrators. They can also work with the developers to identify ways that the recognition of these issues can be included in the initial system requirements and design phases. Understanding how the software and hardware products work and interrelate will help the CSIRT determine the severity, scope, and impact of an incident.

Improving Development Practice

In Essential Factors for Successful Software Security Awareness Training, van Wyk and Steven discuss methods for socializing security experts and developers. They propose a curriculum for educating developers, management, and execu-

tives about security issues with a focus for developers on understanding attacker exploits and corresponding mitigation strategies and methods for executing software security touchpoints within the SDLC [Steven 2006].

CSIRT staff who work in an organization developing software products or designing and implementing enterprise systems would be good candidates to help build such a curriculum within their organization. Their knowledge of incident and vulnerability activity and history within the organization along with their general understanding of security concepts, risks, and threats, their expertise in attack methods and corresponding mitigation strategies, and) their experience with existing organizational systems will allow them to provide real-life, relevant examples to the course attendees.

Another way to create communications channels between CSIRT and development staff is holding joint discussions to review new vulnerabilities and their long-term impacts. A third way may be to specifically assign a member of the CSIRT to participate in design reviews or development work (if time and resources permit.)

Evolutionary Systems Design

CSIRT involvement in the SDLC does not stop with deployment and operations. Once a system is installed, the development cycle still continues. In an article on evolutionary software design, Lipson talks about the need for installed software to be able to adapt to changes in its environment, usage, or components. He introduces the concept of "perpetual design," saying that "all SDLC activities must be perpetual if the quality attributes of a system are to be sustained over time" [Lipson 2006].

Lipson goes on to explain that

> *Any significant change in system requirements can certainly affect the underlying risk management assumptions [for the system], but the effects of other changes might not be as obvious. Therefore, one of the most essential uses for risk management resources would be to support security and survivability monitoring to provide early warnings of emerging threats and increased risks to the system.*

Lipson states that "the first step in evolving to meet new threats to your system's security is to recognize the need for change." Information gathered as a result of recognizing this need for change will be used to institute evolutionary design changes. He then lists the following change factors or triggers that must be monitored as influences on the evolutionary design of secure systems:

- business and organizational

- threat environment
- operating environment
- economic environment and the acquisition marketplace
- political, social, legal, and regulatory environment
- relationships to other systems and infrastructures
- lessons learned and system feedback

Two of the triggers mentioned by Lipson should be familiar by now as obvious areas where CSIRTs can provide business and risk intelligence:

- threat environment
- attack techniques – new and existing techniques used by intruders
- malicious adversaries – changes in attackers such as the rise in new cyber criminals
- lessons learned and system feedback from sources including
- system instrumentation and audits – network monitoring and alerts
- operational experience (attacks, accidents, and failures) – real-life incidents and experiences
- results of periodic security and survivability evaluations – operational exercises, penetration testing, and vulnerability scanning
- technical society meetings, security courses, seminars, journals, news reports – learning from what has happened to others

All of the above sources of change information are related to activities that CSIRT staff may perform, information that they may collect and analyze, or incidents they may receive and respond to. Because CSIRT staff perform these functions on a day-to-day basis, they will also have the most up-to-date information and therefore knowledge on what evolutionary changes might be required. Such knowledge needs to be fed back into the SDLC.

Some not so obvious triggers where CSIRTs may have knowledge that can be useful to identifying evolutionary changes include

- the operating environment. New technologies or trends related to security tools and best practices may provide new techniques for hardening system configurations and preventing or containing attacks.
- the legal or regulatory environment. New requirements for organizations to report security breaches or any unauthorized release of personal privacy information may change the monitoring and alerting requirements for software, change the reporting requirements related to security events and incidents, or mandate response capabilities to be established.1

- the organizational environment and its relationships to other systems and infrastructures. Changes in business practice and user behavior may require significant changes to the threat model used in designing software. Changes can be both technical or socially based. For example, the availability of information can be improved and its confidentiality threatened by both the widespread availability of large-capacity personal storage devices (USB sticks, mobile phones, PDAs, and MP3 players) and the increasing practice of home or remote working. Changes such as moving connectivity from dedicated private lines to communication over the shared Internet should be reflected in software design.

Though not always directly responsible for these areas, CSIRTS are often the groups who learn about such changes first and can pass this information on to other parts of the enterprise. Such information is often gathered through public monitoring of security sites and mailing lists. This type of technology watch function performed by CSIRTs results in information that is important for software developers to know, understand, and eventually synthesize into software and hardware requirements.

SUMMARY

"It will be easier to produce software that is secure if risk management activities and checkpoints are integrated throughout the development life cycle" [Goertzel 2006].

CSIRTs are one source of information and expertise that can provide real insight into current and emerging computer security risk and threats. This experience can be translated into strategies for preventing and responding to computer security incidents. At the most proactive level, such information and corresponding analysis can be fed into the SDLC and used to better define security, alerting, and recovery requirements in systems, hardware, and software.

It is important for CSIRTs—as well as product developers and managers—to understand the role that CSIRTs can play in the SDLC. An effective organization will look to establish methods and communication channels that encourage and support interaction between these two communities. It is also important for these groups to understand that this interaction should be bidirectional: the CSIRT providing risk and threat information and attack explanations, and the product developers helping the CSIRT to understand how software and hardware components and processes work and are intended to be used. CSIRT staff and software developers working together, result in the design and implementation of

better software requirements, which in turn, results in more effective analysis and mitigation of computer security attacks and threats allowing critical business functions to be resilient and successful.

BIBLIOGRAPHY

[Attack Patterns 2006] "Attack Patterns." Build Security In. (2006).

[Barnum 2006a] Barnum, Sean & Sethi, Amit. "Introduction to Attack Patterns." Build Security In. (2006).

[Barnum 2006b] Barnum, Sean & Sethi, Amit. "Attack Pattern Usage." Build Security In. (2006).

[Goertzel 2006] Goertzel, Karen Mercedes et al. Security in the Software Life Cycle, Version 1.2 (Draft). (2006).

[Hope 2005] Hope, Paco; Lavenhar, Steven; & Peterson, Gunnar. "Architectural Risk Analysis." Build Security In. (2005).

[Killcrece 2002] Killcrece, Georgia; Kossakowski, Klaus-Peter; Ruefle, Robin; & Zajicek, Mark. CSIRT Services. (2002).

[Killcrece 2005] Killcrece, Georgia. "Incident Management." Build Security In. (2005).

[Lipson 2006] Lipson, Howard. "Evolutionary Design of Secure Systems – The First Step Is Recognizing the Need for Change." Build Security In. (2006).

[McGraw 2005] McGraw, Gary & van Wyk, Kenneth. "Bridging the Gap between Software Development and Information Security". IEEE Security and Privacy 3, 5 September/October 2005): 75-79.

[Mead 2006a] Mead, Nancy R. "Requirements Elicitation Introduction." Build Security In. (2006).

[Mead 2006b] Mead, Nancy R. "Security Requirements Engineering." Build Security In. (2006).

[Steven 2006] Steven, John; and van Wyk, Kenneth R. "Essential Factors for Successful Software Security Awareness Training," IEEE Security & Privacy 4, 5 (September/October 2006): 80-83.

[van Wyk 2006] van Wyk, Kenneth. "Adapting Penetration Testing for Software Development Purposes." Build Security In.

[West Brown 2003] West Brown, Moira J..; Stikvoort, Don; Kossakowski, Klaus Peter.; Killcrece, Georgia; Ruefle, Robin; & Zajicek, Mark. Handbook for Computer Security Incident Response Teams (CSIRTs) (CMU/SEI-2003-HB-002, ADA413778). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM-0001120