Maintenance

Objective

To ensure that

* products are properly maintained post deployment
* minimize the attack surface area through out the production lifecycle
* security defects are fixed properly and in a timely fashion

Platforms Affected

All.

Relevant COBIT Topics

DS6 – Manage Changes – All sections should be reviewed

Best Practices

There is a strong inertia to resist patching “working” (but vulnerable) systems. It is your responsibility as a developer to ensure that the user is as safe as is possible and encourage patching vulnerable systems rapidly by ensuring that your patches are comprehensive (ie no more fixes of this type are likely), no regression of previous issues (ie fixes stay fixed), and stable (ie you have performed adequate testing).

Supported applications should be regularly maintained, looking for new methods to obviate security controls

It is normal within the industry to provide support for n-1 to n-2 versions, so some form of source revision control, such as CVS, ClearCase, or SubVersion will be required to manage security bug fixes to avoid regression errors

Updates should be provided in a secure fashion, either by digitally signing packages, or using a message digest which is known to be relatively free from collisions

Support policy for security fixes should be clearly communicated to users, to ensure users are aware of which versions are supported for security fixes and when products are due to be end of lifed.

Security Incident Response

Many organizations are simply not prepared for public disclosure of security vulnerabilities. There are several categories of disclosure:

* Hidden
* 0day
* Full disclosure and limited disclosure
* With and without vendor response

Vendors with a good record of security fixes will often gain early insight into security vulnerabilities. Others will have many public vulnerabilities published to 0day boards or mailing lists.

How to determine if you are vulnerable

Does the organization:

* Have an incident management policy?
* Monitor abuse@...
* Monitor Bugtraq and similar mail lists for their own product
* Publish a security section on their web site? If so, does it have the ability to submit a security incident? In a secure fashion (such as exchange of PGP keys or via SSL)?
* Could even the most serious of security breaches be fixed within 30 days? If no, what would it take to remedy the situation?

If any of the questions are “no”, then the organization is at risk from 0day exposure.

How to protect yourself

* Create and maintain an incident management policy
* Monitor abuse@...
* Monitor Bugtraq and similar mail lists. Use the experience of similar products to learn from their mistakes and fix them before they are found in your own products
* Publish a security section on their web site, with the ability to submit a security incident in a secure fashion (such as exchange of PGP keys or via SSL)
* Have a method of getting security fixes turned around quickly, certainly fully tested within 30 days.

Fix Security Issues Correctly

Security vulnerabilities exist in all software. Occasionally, these will be discovered by outsiders such as security researchers or customers, but more often than not, the issues will be found whilst working on the next version.

Security vulnerabilities are “patterned” – it is extraordinarily unlikely that a single vulnerability is the only vulnerability of its type. It is vital that all similar vulnerabilities are eliminated by using root cause analysis and attack surface area reduction occurs. This will require a comprehensive search of the application for “like” vulnerabilities to ensure that no repeats of the current vulnerability crop up.

Microsoft estimates that each fix costs more than $100,000 to develop, test, and deploy, and obviously many tens of millions more by its customers to apply. Only by reducing the number of fixes can this cost be reduced. It is far cheaper to spend a little more time and throw a little more resources at the vulnerability to close it off permanently.

How to identify if you are vulnerable

Certain applications will have multiple vulnerabilities of a similar nature released publicly on mail lists such as Bugtraq. Such applications have not been reviewed to find all similar vulnerabilities or to fix the root cause of the issue.

How to protect yourself

* Ensure that root cause analysis is used to identify the underlying reason for the defect
* Use attack surface area reduction and risk methodologies to remove as many vulnerabilities of this type as is possible within the prescribed time frame or budget

Update Notifications

Often users will obtain a product and never upgrade it. However, sometimes it is necessary for the product to be updated to protect against known security vulnerabilities.

How to identify if you are vulnerable

* Is there a method of notifying the owners / operators / system administrators of the application that there is a newer version available?

How to protect yourself

Preferably, the application should have the ability to “phone home” to check for newer versions and alert system administrators when new versions are available. If this is not possible, for example, in highly protected environments where “phone home” features are not allowed, another method should be offered to keep the administrators up to date.

Regularly check permissions

Applications are at the mercy of system administrators who are often fallible. Applications that rely upon certain resources being protected should take steps to ensure that these resources are not publicly exposed and have sufficient protection as per their risk to the application.

How to identify if you are vulnerable

* Does the application require certain files to be “safe” from public exposure? For example, many J2EE applications are reliant upon web.xml to be read only for the servlet container to protect against local users reading infrastructure credentials. PHP applications often have a file called “config.php” which contains similar details.
* If such a resource exists, does relaxing the permissions expose the application to vulnerability from local or remote users?

How to protect yourself

The application should regularly review the permissions of key files, directories and resources that contain application secrets to ensure that permissions have not been relaxed. If the permissions expose an immediate danger, the application should stop functioning until the issue is fixed, otherwise, notifying or alerting the administrator should be sufficient.