G Data
TechPaper #0271

Patch Management Best Practices

G Data Development

TechPaper_#0271_2013_05_07

G Data. Security Made in Germany.

Contents

1 Introduction

The increased complexity of enterprise networks and the ever-present threat of malware present a challenge for every network and system administrator. Not only has the number of software installations to be patched risen significantly, the speed at which vulnerabilities are being exploited has also strongly increased. To deal with the task of patch installation, specialized patch management systems perform automated tasks and ensure timely deployment of security-related fixes. To develop a full workflow around patch management, including full control of network assets and software, enterprises need to think of more than just software to manage deployment. To assist in effectively running a patch management procedure, this document will outline standards in patch management, as well as recommendations and best practices for small businesses (up to 50 network clients) as well as enterprises (50+ network clients).

1.1 Definition

Even though many vendors strive for perfection at release time, most software products will need to be serviced with updates and patches during their lifetime. Typically, updates provide new functionality or better performance, while patches fix software bugs. The former category is usually not of critical importance for enterprise deployment, but the latter requires swift action: especially in the case of security issues, patches should quickly be deployed across the corporate network to prevent possible exploitation.

Patch management aims to streamline deployment of patches. Updates are often included in the process, making use of the technical and organizational infrastructure that is being set up to create a unified update/patch management system (UPMS). A complete UPMS comprises more than just the technical possibilities to deploy patches across the network. The time spent on actual deployment should be minimized to focus available resources on recognizing, classifying and remediating security issues. Depending on the size of the organization, this may require dedicated personnel or at the very least workflow procedures to ensure speedy decision making in case of security emergencies.

Patch management procedures should be used in any company where the integrity and security of the computer network need to be managed efficiently. This goes for small business networks as much as for large enterprise networks. Centralizing patch management helps establishing a security baseline for the whole network and facilitates simple and swift patch deployment.

1.2 Significance

Patches often repair security vulnerabilities through which attackers may gain access to systems running the affected software. In responding to security emergencies, rapid deployment of patches is important. A complicating factor, the release of a patch actually stimulates hackers to develop and exploit the security bug, due to the public release of information about the vulnerability that typically accompanies patch releases. By reverse-engineering patch files, attackers can obtain the information necessary to stage an effective attack. This puts extra pressure on administrators to timely patch their systems. Patch management helps speed up patch deployment and improves the efficiency of the complete process by coordinating and standardizing patch deployment procedures.

Not applying patches leaves systems exposed. Attackers who take advantage of software bugs can, depending on the severity of the issue, gain access to files saved on a PC, execute programs, take over other PCs in the corporate network, or worse. While a malware infection through a drive-by download or an attack from hackers are annoying for home users, corporate networks are especially vulnerable. The stakes are much higher: the mere presence of a hacker in the company network compromises data integrity, and can lead to loss of data if one or more systems are irreparably damaged. Further threats include downtime for critical systems, intellectual property theft, loss of reputation, or excessive costs for legal defense if customer data is lost.

Standardized patching procedures help prevent successful exploitation of software bugs by hackers. However, patch management is not the only measure that should be taken. Even if software is fully patched, hackers may be aware of bugs that have not been discovered yet by the software vendors. Business networks and clients should always be protected by security products on a client level, providing measures such as signature based malware recognition, heuristic scanning and file reputation management, as well as protection on the network level.

1.3 Compliance

Regulations for patch management as an independent process rarely exist. For many companies, patch management is part of a wider array of measures taken in the context of information security. This field is well documented and many companies already comply with the applicable standards, most notably ISO/IEC 27002:2005 (due to be revised at the end of this year)[1]. This standard, which has been adapted by many national standards bodies, establishes guidelines for all aspects of organizational information management, and presents standards for a complete information security management system. Similarly, the Information Security Forum's Standard of Good Practice is a best-practices based guide to information security[2]. Additionally, ISO standard 15408-1:2009, also known as Common Criteria for Information Technology Security Evaluation (CC), provides a framework to specify, implement and test security requirements[3].

On a more practical level, government agencies in several countries have published their own standards and recommended practices regarding patch management. Different guideline documents apply to different sectors of the economy. Private enterprises in the United States, for example, can consult the expertise of the National Institute of Standards and Technology's Guide to Enterprise Patch Management Technologies (SP 800-40 Rev. 3 Draft)[4]. Critical infrastructure providers, including federal organizations, heavy industry and military alike, are served by the National Cyber Security Division of the Department of Homeland Security. In 2008 this institution published the document Recommended Practice for Patch Management of Control Systems, focusing on critical infrastructure[5]. Europe's national governments have undertaken similar endeavors. The United Kingdom Centre for the Protection of National Infrastructure provides the Good Practice Guide Patch Management, again aimed at critical national infrastructure organizations[6]. In Germany, the Federal Office for Information Security (BSI) provides advice to small companies as well as large enterprises, as part of a wider change management advisory[7].

[1] See www.iso.org/iso/catalogue_detail?csnumber=50297.
[2] See www.securityforum.org/downloadresearch/publicdownload2012sogp.
[3] See www.commoncriteriaportal.org/cc.
[4] See csrc.nist.gov/publications/PubsSPs.html.
[5] See ics-cert.us-cert.gov/practices/documents/PatchManagementRecommendedPractice_Final.pdf.
[6] See www.cpni.gov.uk/Documents/Publications/2006/2006029-GPG_Patch_management.pdf.
[7] Good starting points include www.bsi.bund.de/ContentBSI/grundschutz/kataloge/baust/b01/b01014.html for enterprise administrators or www.bsi.bund.de/ContentBSI/grundschutz/kataloge/m/m02/m02221.html for SMB.

2 Patch management

Patch management is important for each computer, be it a home device or corporate workstation. In any case, the availability of new security patches should be actively monitored and fixes should be deployed as soon as possible. However, the way in which patches are managed and deployed is mostly a question of scale. For home users, Microsoft's built-in Windows Update can provide security fixes for Windows in a completely automated way, and more vendors are moving towards a fully transparent, automatic updating process (Adobe with its Adobe Reader and Adobe Flash Player software, as well as browsers like Mozilla Firefox and Google Chrome). They are regularly alerted by built-in updaters of other software. Tech-savvy users may choose to use commercial update notification software, to make sure no patches are being skipped[8]. Nevertheless, home PCs are among the most insecure devices in regards to patch management, mostly due to indifference or a lacking sense of urgency on the side of the end user. The complexity of keeping an overview of installed software, its vulnerabilities, and its patches is already overwhelming for single PC users – let alone administrators of business networks with any number from five up to thousands of clients. This is where a standardized, recurring patch management procedure can help, by reducing the time required to take an inventory of software and vulnerabilities, and by automating the deployment. An effective patch management procedure clarifies which responsibilities lay with whom, tracks all changes that are being made, provides a rollback method, tests all proposed changes extensively, and announces changes to all involved parties.

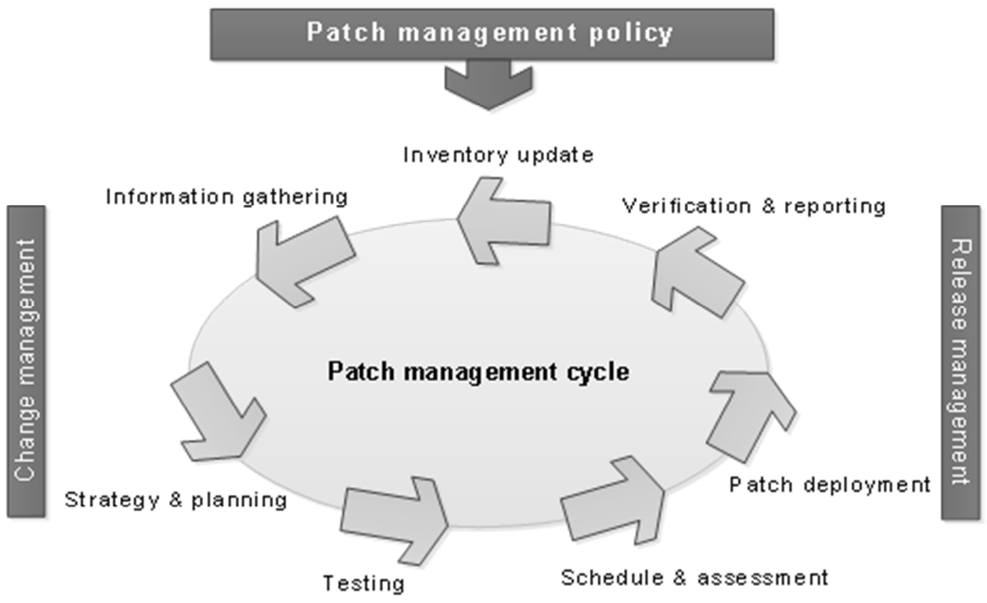


***Image 1**: Patch management cycle*

The patch management cycle can be broken down into different stages (which will be discussed in detail in chapter 3). Note that it is a cycle, not an event-driven process. Patches should be proactively deployed, therefore patch management should be proactively carried out. Each step of the cyclical procedure should be explicitly defined and assigned to someone. Depending on their wishes and

[8] Several commercial parties offer update notification software, for home users as well as small businesses. Some of these solutions only function as reminders, others almost resemble patch management solutions. See, for example, UpdateStar at www.updatestar.com.

requirements, companies can merge stages by bundling them and assigning them to the same person, or define further action points as required. Existing change management and release management standards can be (partially) integrated. Some steps of the procedure can be automated, most notably the deployment, but several key actions will have to be carried out manually for each cycle. To optimize this process, planning is crucial.

The rate of recurrence of the patch management cycle depends on the resources that can be made available to execute it, as well as on the rate with which patches are published for software that is in use in the network. A major source of patches is Microsoft, whose Windows operating system is served with the latest security updates and improvements monthly. Some other vendors have chosen to align their update cycle with Microsoft, most notably Adobe (whose Adobe Reader and Flash Player software are along the most widely deployed installations)[9]. These predictable patch cycles are scheduled to occur on a fixed date, in the case of Microsoft and Adobe every second Tuesday of the month. This allows system administrators to plan a recurring, reliable procedure to roll out the latest updates every month. Other vendors release updates less frequently, such as Oracle, which has chosen to distribute security patch bundles for the widely used Java platform every three to four months[10]. Some vendors choose to release patches whenever a critical issue comes up. For patch management planning, this quick response time comes at a price: it may take some time before an unpredictably released patch is thoroughly tested and deployed. On the other hand, rigid patch bundling can cause some security holes to go unpatched for a significant time until the next patch release date.

2.1 Patch management policy

Before planning the monthly steps for a patch management cycle and assigning responsibilities, some standards have to be defined. A patch management policy helps decision making during the cycle. The policy should cover questions about patching strategy. Should all available patches be installed by default, or will there be a classification, possibly based on the severity of the security issues they remedy? Will patches be installed proactively (to plug possible security holes) or reactively (only when problems arise), or a combination of both? To prevent spending unnecessary time on patch-by-patch decisions, it is recommended to set as many generalized rules as possible. At the same time, simply installing every available patch is not a solution: to prevent network and system load and compatibility problems, conscious choices need to be made.

Other parameters that the patch management cycle depends on include installation standards, network standards, and application security configuration. While each cycle includes making an inventory of the current state of the network, effort is greatly reduced when the network environment has been set up following a standard. Define which software installations are allowed, and which machines are provided with which software. Using whitelists or blacklists, the network software inventory can be limited, greatly helping patch deployment. The same goes for security configuration, user accounts and passwords.

By standardizing policies across the network, there will be fewer exceptions to be handled during patch deployment. However, it can still be fruitful to define actions to undertake in case exceptions do occur. Administrators should not be surprised by machines that turn out to be unreachable at

[9] Adobe releases quarterly and out-of-band updates for its Acrobat and Reader products (helpx.adobe.com/acrobat/release-note/release-notes-acrobat-reader.html). Its Flash Player has recently been moved to a rapid release cycle (blogs.adobe.com/flashplayer/2012/09/flash-runtime-rapid-release-cycle.html).
[10] See www.oracle.com/technetwork/topics/security/alerts-086861.html.

deployment time, patches that cause compatibility problems during testing or deployment, or security issues that do not seem to have been mitigated successfully after patching. Swift redeployment, software reconfiguration and incident escalation should be defined as part of the patch management policy.

An important aspect to keep in mind is the deployment of new machines in a corporate network. While outside the strict scope of patch management, it is important to make sure that system images with which new clients are deployed are maintained with the latest patches. If outdated systems get deployed, there is a vulnerability window that will only be closed once the newly deployed machine has become part of the patch management cycle and its inventory has been listed.

2.2 Enterprise versus small businesses (SMB)

Business clients require more control over how and when which patches are installed than home users. Using the standard update mechanisms for each installed product leads to a chaotic network state, where some clients are running other software versions than others. Centrally administering updates benefits every business network, by reducing the impact on usability and ensuring a speedy mitigation of security issues. However, not all business networks are created equally. Patch management for enterprise networks differs from smaller SMB networks.

For enterprise patch management, it is important to make a proper assessment of the network organization and zones, and of the different roles of network clients. Depending on the size of the network, there can be several client groups, each of which has its own distinct configuration. Some groups can be served with almost every patch available, while others need to be tested more carefully. Consider the following example:

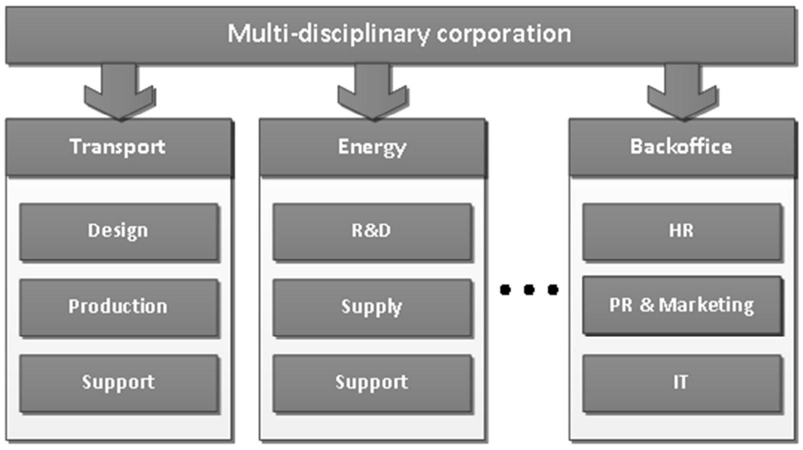


***Image 2**: Enterprise network organization*

A corporation with multiple business groups across several disciplines usually cannot apply the same patches policy to all groups. The different software packages that are in use across groups are one issue, but especially the impact that patches can have on work environments is problematic. While generic client configurations (Microsoft Office, browser), which are often used for backoffice client roles, can easily be patched, some business groups may need a work environment that does not change. Clients that are used in a Quality Assurance role or similar functions that depend on unchanging environments will have to be carefully integrated into the UPMS, or manually serviced.

Lack of manpower and budget are the main challenges preventing SMBs from running a full-time patch management procedure monitoring the latest security issues in near real-time. At the same time, they are a major target for cyber criminals, who are aware of the lack of attention to security and lack of budget for sophisticated security tools. Therefore, patch management for small businesses, while based on the same cycle as the enterprise procedure, has a few key differences. To make sure that the procedure can also be carried out in networks where there is no budget or manpower for permanent protection, many measures can be scaled down (provided the network has not too many clients – around fifty, for a typical SMB network).

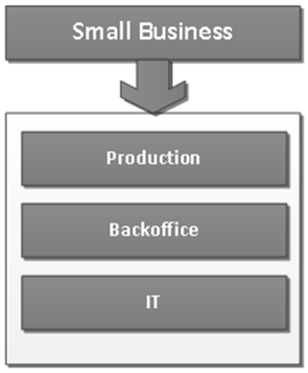


***Image 3**: SMB network organization*

Patching strategy and testing are simplified enormously when the number of scenarios to be tested is reduced. Smaller companies have significantly less business groups, and therefore less diversity in network client roles. Still, as with enterprise patch management, it is essential to spend some time to produce an abstract overview of the network and the types of clients running in it. The diagram shows a company with only one business group. An SMB with only one production specialization can divide its network clients into just two or three roles. The variety of patches to be installed is a lot smaller and even though no steps of the patch management cycle are being omitted, they take considerably less time. Further efficiency increases can be achieved by extensively standardizing software deployments across the network. Especially for smaller companies that might not have a dedicated system or network administrator, streamlining the procedure is important. SMBs can use a patch management solution to automate almost the whole cycle. Keeping up to date with the latest exploit information and software versions can require a lot of time; a patch management solution can help those companies that do not have the manpower to continuously monitor their network security status. An alternative is to completely outsource IT security, including patch management, to a service provider that will take care of the IT infrastructure as a managed service.

As a helpful tool, small businesses can create a checklist or todo-list to serve as a guideline to patch management. Based on the patch management cycle, a checklist provides a way to run through the most important actions relatively quickly. For SMBs, it can even replace the expansive patch management policy, but it still needs to define many of the points that the policy calls attention to:

Inventory update
- ✓ List installed products across the network and their vendors
- ✓ Find out the patch cycle for each vendor
- ✓ Prioritize products by importance

Information gathering
- ✓ Check for new patches once per patch cycle for each vendor

Testing
- ✓ Test the patches for all applicable systems

Deployment
- ✓ Deploy patches and verify deployment

3 Patch management procedure

3.1 Step I: Inventory update

The first step of the patch management cycle is to update the inventory. For each cycle, this overview needs to be updated and complemented with statistics such as version information for all machines. To make sure all machines are included in the inventory, several methods may be useful. Many enterprise networks using Windows Server make use of Active Directory services. The domain controller can easily produce a list of all clients that are available in the network domain(s). However, this still leaves out a potentially large number of machines that have not joined the Windows domain. A lower-level list of network clients can be obtained by accessing the log files of the network's DHCP server (IP address or subnet scan) or by checking the local DNS registrations. A combination of these methods will produce the best results, as not all machines will be listed by every method. Scans at multiple points in time can be necessary to obtain a full list. For smaller networks, obtaining an overview of clients should not be problematic. If Windows File and Printer Sharing has been enabled, the machines should show up in the Network Places window. Especially complicated is the presence of virtual machines within the network. Virtualized network clients may be offline during inventory or deployment. Still, listing them as integral part of the network and patching them accordingly is vital, because each virtual machine represents a potential entry point into the network.

As part of the patch management strategy, network clients and their configuration should be deployed following certain standards, making the production of a full software list relatively easy. Deploying only software that is absolutely required decreases the likelihood of security holes being present in the system and makes patch management less time-intensive. Monitoring software installations, or completely blocking non-approved software, greatly reduces the amount of time spent on identifying, obtaining and distributing patches. Using a software policy blacklist or whitelist is an efficient way to limit new software installations. Moreover, end users should not be able to modify patch and updater settings for their local software installations to prevent discrepancies between local deployments and server policies. This includes disabling auto-updaters for software such as Flash Player and Adobe Reader, as well as stand-alone Windows Update services. Carefully check all deployed software for their update policies and make sure the functions are taken care of centrally.

Sets of information that should be obtained as part of the inventory include version information of the operating system that is being used and a full list of software and patches. This data is the very minimum that need to be collected for every network client, to enable easy inquiries about available patches. Along similar lines, collecting information about the hardware in use can help prevent or circumvent problems like a lack of disk space or CPU power to process patches. And as with software, a baseline hardware deployment greatly simplifies the identification process and makes the whole procedure more predictable.

To make sure that the actual deployment can be carried out without problems, take note of all services running on the network clients. During the initial client deployment, these should already have been configured for maximal performance and minimal vulnerability: the less services that are running on a client, the less likely it is that one of them can be used as an attack vector. During the inventory phase, superfluous services can be identified and disabled to make sure that none of them are interfering with the patching process. Similarly, make sure that each client can be accessed with the correct permissions required to install software. Without administrator access, the patching process cannot be carried out successfully. Finally, ensure that all machines have proper, uninterrupted network access at a decent speed. Problems with network connectivity, either by network overload or local issues, can cause significant amounts of errors that later need to be taken care of manually, or substantially slow down patch deployment.

Various techniques can be used to update the inventory, depending on the data that need to be collected. A relatively non-intrusive way is to perform agentless information gathering. This centralized method lets the server pull information from network clients without having to install agent software in each machine. In order to probe the machines, the server needs to have the correct permissions. The centralized layout allows for a relatively easy setup procedure: only the server needs to be configured. Machines that are newly added to the network can be scanned, even without a local agent. The downside is the lack of support for machines that are not always connected to the enterprise network, or not always turned on. Additionally, network security software can block agentless scanning techniques. Agent-based scanning circumvents these issues. Agents connect to the server as soon as the clients are powered on. No network-wide scanning schedule (including the bandwidth use this incurs) is required. A huge advantage over agentless scanning methods is the integration of patch management with existing agent-based security solutions. Information that is gathered in the context of the regular protection can easily be repurposed for the patch management cycle. As a separate technique, network monitoring can be used to identify software by its network traffic. For platforms where agents are not supported, this can be a useful additional tool, but passive network monitoring is too unreliable to use as a main component. The technique itself, however, can be quite useful: many software vulnerabilities can be detected by network-based vulnerability scanners. Running a vulnerability scan as part of the inventory phase helps in quickly locating security issues that need to be patched (but does not always provide information about patch availability or workarounds).

Not all clients can be included in the patch management cycle. While the greatest effort should be made to provide all machines with the latest updates, there are valid reasons for excluding some clients. Some business groups may need unchanging environments (for testing, evaluation and comparison purposes), requiring extra tests before patch deployment. Similarly, legacy clients will need extra attention. Some legacy software may require older operating systems or software which fall outside the software baseline or are not maintained anymore. Patching such systems can break system critical applications so extra testing is in order. Some legacy software can require a separate vendor patch to add compatibility with patched versions of other software. It is important to discover this kind of requirements as early as possible. If the inventory phase indicates that legacy software is being run, there is a higher possibility of compatibility issues.

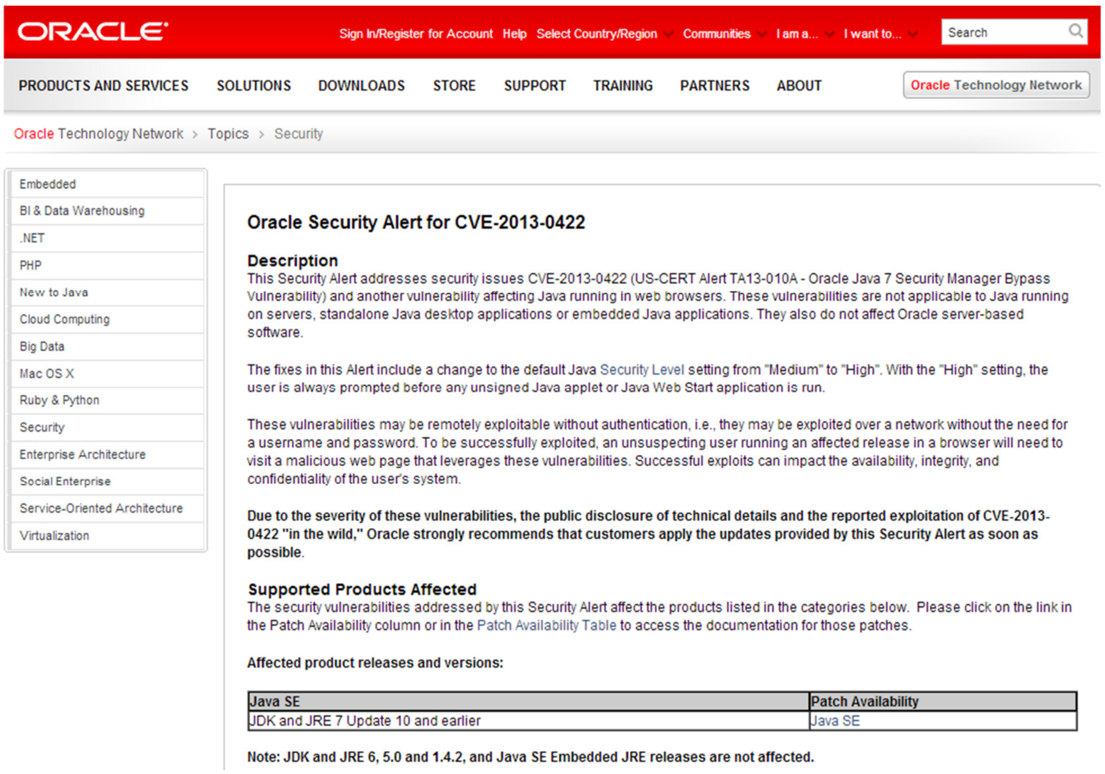
Clients that are not always connected to the network, such as laptops for personnel on the road, or VPN clients, can still be patched during the regular cycle, but they might not connect to the network until well after the scheduled inventory or deployment time. Make sure that network and machine load do not spike upon first logon; spreading inventory and deployment jobs over time for this type of clients is advisable. Virtual machines that are running directly in the company's network should also be managed, if they are not separated from the regular production environment by security appliances or VLAN configuration.

In general, machines that for whichever reason are not (yet) included in the patch management cycle, or have outdated software installations, should be tested extra carefully for security holes and malware infections. Scheduling extra malware scans or setting up a separate network or firewall zone helps prevent problems. Using security policies, access to network resources can be restricted.

Blocking removable media from being used can prevent eventual malware infections from spreading across the network.

3.2 Step II: Information gathering

As soon as a full inventory has been produced or updated, it is important to keep tabs on new releases of software updates and patches, as well as exploits and other possible security issues. For every product deployed on network clients, the administrator should ideally always know at which version it is, if there are any known bugs or security holes, and whether patches or updates are available. This is important for both enterprise and SMB networks: even if an SMB does not have budget to appoint a full-time systems administrator, information gathering is the basis of a responsible patch management procedure. Relying on a third party aggregator, such as news websites, or patch management solutions that offer notifications for new patches, can save time.


ORACLE
Sign In/Register for Account Help Select Country/Region Communities I am a... I want to...
Search
PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT
Oracle Technology Network
Oracle Technology Network > Topics > Security
Embedded
BI & Data Warehousing
.NET
PHP
New to Java
Cloud Computing
Big Data
Mac OS X
Ruby & Python
Security
Enterprise Architecture
Social Enterprise
Service-Oriented Architecture
Virtualization

Oracle Security Alert for CVE-2013-0422

Description
This Security Alert addresses security issues CVE-2013-0422 (US-CERT Alert TA13-010A - Oracle Java 7 Security Manager Bypass Vulnerability) and another vulnerability affecting Java running in web browsers. These vulnerabilities are not applicable to Java running on servers, standalone Java desktop applications or embedded Java applications. They also do not affect Oracle server-based software.

The fixes in this Alert include a change to the default Java Security Level setting from "Medium" to "High". With the "High" setting, the user is always prompted before any unsigned Java applet or Java Web Start application is run.

These vulnerabilities may be remotely exploitable without authentication, i.e., they may be exploited over a network without the need for a username and password. To be successfully exploited, an unsuspecting user running an affected release in a browser will need to visit a malicious web page that leverages these vulnerabilities. Successful exploits can impact the availability, integrity, and confidentiality of the user's system.

Due to the severity of these vulnerabilities, the public disclosure of technical details and the reported exploitation of CVE-2013-0422 "in the wild," Oracle strongly recommends that customers apply the updates provided by this Security Alert as soon as possible.

Supported Products Affected
The security vulnerabilities addressed by this Security Alert affect the products listed in the categories below. Please click on the link in the Patch Availability column or in the Patch Availability Table to access the documentation for those patches.

Affected product releases and versions:

Java SE	Patch Availability
JDK and JRE 7 Update 10 and earlier	Java SE

Note: JDK and JRE 6, 5.0 and 1.4.2, and Java SE Embedded JRE releases are not affected.

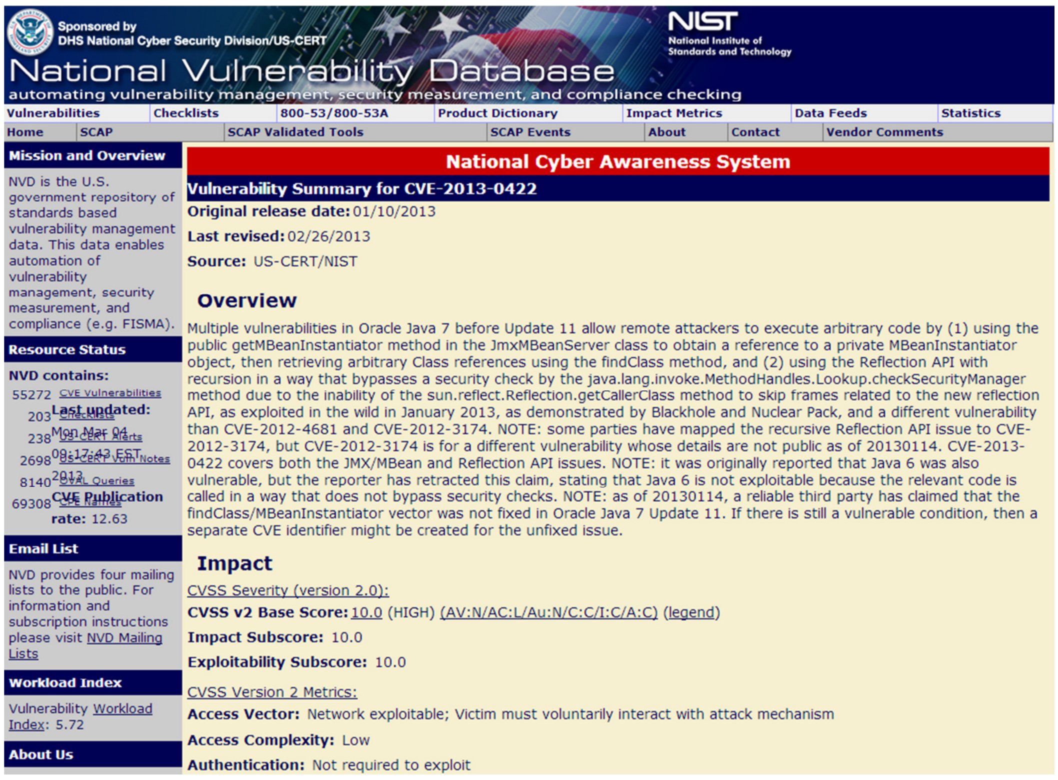
Image 4*: Vendor security advisory*[11]

A very basic method of checking for updates is contacting the respective vendors. Most software vendors publish software updates on their website, often including release notes detailing the latest fixes and additions. Some websites have RSS or e-mail notification services available, a low-cost and highly effective way to stay up to date. While getting information directly from the source is a very reliable way to ensure authenticity, having to check for updates for each product separately involves a lot of unnecessary work. Many patch management software solutions maintain their own database of version information, allowing administrators to quickly compare their inventory to the latest available patch information. While there is a risk in depending on a third party for information about

[11] See http://www.oracle.com/technetwork/topics/security/alert-cve-2013-0422-1896849.html.

updates, because as an administrator you would not be getting information directly from the vendor, the use of a third party database greatly simplifies the process of information gathering. Third party databases offer information of a higher quality, often being enriched with additional patch classifications, verification of patches and documentation about compatibility with well-known business software products.

While information about the patches themselves is vital for managing a software environment, it does not cover the whole spectrum. Before a patch is released, there are several stages of patch development during which it is already known that a patch is forthcoming. When a software vendor becomes aware of a security issue, they often publish a security advisory, as pictured above. An advisory usually contains details about the severity of the issue, as well as a timeline for patch development. Temporary workarounds may be available to allow administrators to mitigate the issue before the vendor releases the patch.


Sponsored by
DHS National Cyber Security Division/US-CERT
National Vulnerability Database
automating vulnerability management, security measurement, and compliance checking
NIST
National Institute of Standards and Technology
Vulnerabilities | Checklists | 800-53/800-53A | Product Dictionary | Impact Metrics | Data Feeds | Statistics
Home | SCAP | SCAP Validated Tools | SCAP Events | About | Contact | Vendor Comments

Mission and Overview

NVD is the U.S. government repository of standards based vulnerability management data. This data enables automation of vulnerability management, security measurement, and compliance (e.g. FISMA).

Resource Status

NVD contains:
55272 CVE Vulnerabilities
203 Checklists
238 US-CERT Alerts
2698 US-CERT Vuln Notes
8140 OVAL Queries
69308 CPE Names

Last updated: Mon Mar 04 09:17:43 EST 2013

CVE Publication rate: 12.63

Email List

NVD provides four mailing lists to the public. For information and subscription instructions please visit NVD Mailing Lists

Workload Index

Vulnerability Workload Index: 5.72

About Us

National Cyber Awareness System

Vulnerability Summary for CVE-2013-0422

Original release date: 01/10/2013

Last revised: 02/26/2013

Source: US-CERT/NIST

Overview

Multiple vulnerabilities in Oracle Java 7 before Update 11 allow remote attackers to execute arbitrary code by (1) using the public getMBeanInstantiator method in the JmxMBeanServer class to obtain a reference to a private MBeanInstantiator object, then retrieving arbitrary Class references using the findClass method, and (2) using the Reflection API with recursion in a way that bypasses a security check by the java.lang.invoke.MethodHandles.Lookup.checkSecurityManager method due to the inability of the sun.reflect.Reflection.getCallerClass method to skip frames related to the new reflection API, as exploited in the wild in January 2013, as demonstrated by Blackhole and Nuclear Pack, and a different vulnerability than CVE-2012-4681 and CVE-2012-3174. NOTE: some parties have mapped the recursive Reflection API issue to CVE-2012-3174, but CVE-2012-3174 is for a different vulnerability whose details are not public as of 20130114. CVE-2013-0422 covers both the JMX/MBean and Reflection API issues. NOTE: it was originally reported that Java 6 was also vulnerable, but the reporter has retracted this claim, stating that Java 6 is not exploitable because the relevant code is called in a way that does not bypass security checks. NOTE: as of 20130114, a reliable third party has claimed that the findClass/MBeanInstantiator vector was not fixed in Oracle Java 7 Update 11. If there is still a vulnerable condition, then a separate CVE identifier might be created for the unfixed issue.

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 10.0 (HIGH) (AV:N/AC:L/Au:N/C:C/I:C/A:C) (legend)

Impact Subscore: 10.0

Exploitability Subscore: 10.0

CVSS Version 2 Metrics:

Access Vector: Network exploitable; Victim must voluntarily interact with attack mechanism

Access Complexity: Low

Authentication: Not required to exploit

***Image 5**: Common Vulnerabilities and Exposures (CVE) database entry*[12]

Vendors can choose to notify a CVE Numbering Authority, which manages the database of Common Vulnerabilities and Exposures (CVE). Many vulnerabilities are assigned a CVE number, enabling easy communication about the issue even before a patch is being released. The central CVE database can help administrators in keeping track of software vulnerabilities that will need to be patched at some point. At the moment, the central database is being maintained by the United States National Institute of Standards and Technology (NIST)[13], of which example output is pictured above. Although the CVE database is a widely used central repository, not all security holes are submitted there.

[12] See http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-0422.
[13] See nvd.nist.gov.

In spite of some vulnerabilities never making it into the CVE database, it still produces a sizable daily output of information. For most administrators, tracking all newly assigned CVE numbers will prove too labor-intensive. Independent sites as well as government-sponsored computer emergency response teams (CERTs) are useful sources of information. The United States Department of Homeland Security, for example, posts vulnerability updates and background information at its homepage[14]. Europe features national bodies as well as the centrally coordinated CERT-EU[15] and the European Network and Information Security Agency[16]. Most of these websites provide security advice and hourly or daily news updates.

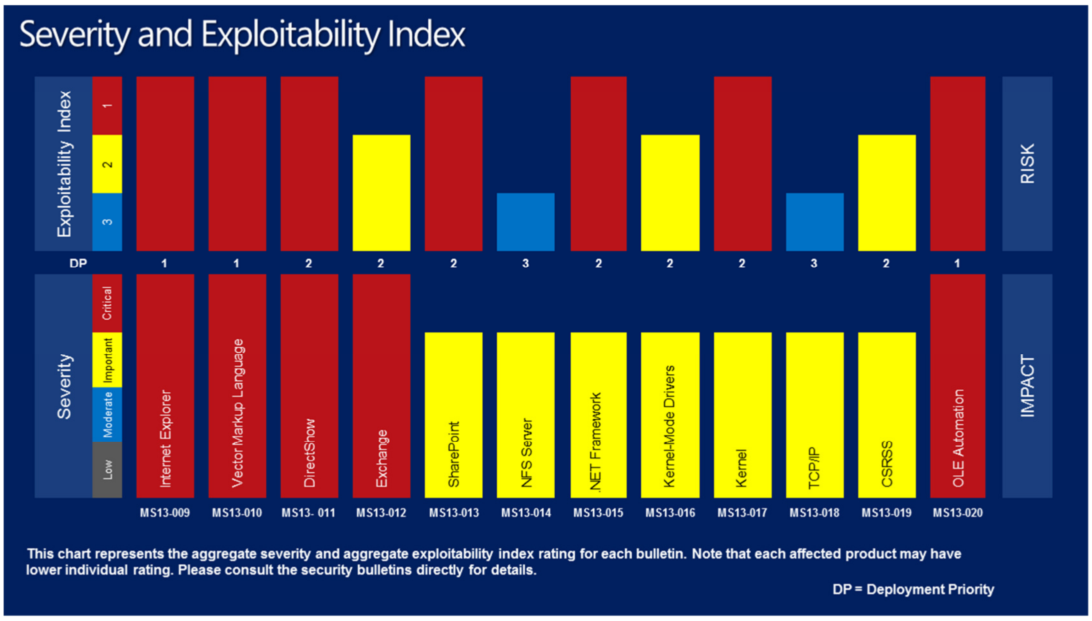
Both vendor information and CVE database listings provide relatively technical information, based on incidents. CERT information is usually more accessible, and can also be instructive to stay informed about general trends in malware. Several media outlets offer online articles about aspects of computer security. Antivirus vendors themselves often release informative whitepapers or maintain blogs covering the latest threats.

3.3 Step III: Strategy and planning

As soon as information about (soon to be) released patches and updates has been obtained, the strategy and planning phase starts. Many questions that are to be answered during this step should already have been taken into account in the patch management policy. For example, it is important to choose which patches to install first, or at all. Based on the severity of the security hole, it may be necessary to initiate a faster deployment than usual, or to deploy a quick workaround.

The first vital realization is that not all available patches and updates need to be installed. The company's patch management policy should define which types of patches and updates will be installed on the network, to prevent complicated questions from slowing down the decision process. For example, while a vendor may deem a certain update necessary, a company may not need the specific function that it adds. Vendors may even choose to remove features that an enterprise depends on. Ignoring the update from an early stage saves time by avoiding further consideration, deployment efforts, and management, so administrators can allocate resources to relevant updates and patches. In some circumstances, even security patches can be ignored if they fix a security hole that cannot be abused in the context of the enterprise deployment of the software. However, in this case, a "better safe than sorry" policy is recommended.

[14] See www.us-cert.gov.
[15] See cert.europa.eu.
[16] See www.enisa.europa.eu.



This chart represents the aggregate severity and aggregate exploitability index rating for each bulletin. Note that each affected product may have lower individual rating. Please consult the security bulletins directly for details.

***Image 6**: Microsoft's Severity and Exploitability Index*[17]

Regardless of the source, information about security issues or patches will almost always provide a severity rating, which helps decide whether to deploy the patch or not, and if so, how quickly. Either the vendor themselves or a relevant agency rates the severity of the security hole. The image above shows a typical severity and exploitability index as published by Microsoft. It details the risk of all security holes that are being patched in a certain month, as well as the impact that an attack on the vulnerable component could have. The severity rating helps decide which issues to patch first, if they apply to a product that is in use on the company network.

The more severe the security hole, the quicker it should be plugged by deploying a patch. The severity rating depends on several parameters. If an attacker can abuse the vulnerability only locally, with physical access to the PC on which the software is installed, the severity would be rated lower than an attack which can be carried out remotely (i.e. via the internet). Through some attacks, hackers can only crash the affected software. Other issues allow them to read or write data to the file system or to launch any program – allowing data to be stolen or connected devices to be compromised. These severe security holes should be patched as soon as possible. Another factor is the availability of knowledge about the vulnerability. If it has been discovered by the vendor, there is only a small chance that it is actively being abused by hackers, who would have had to separately discover the same flaw. However, if the security hole has been publicly disclosed by security researchers, or if the vendor has acquired information about the issue from a vulnerability trading company, many more parties may have access to information about the flaw and successfully built attacks for it. In this case, vendors will have to proceed with patch publication as soon as possible, and deployment should be planned accordingly. A complication is the fact that a patch release often signals the start of a period of intense attacks: hackers reverse-engineer the patch to find out how the vulnerability works and try to exploit unpatched systems as quickly as possible.

In any case, the network client roles that have been laid out in the patch management policy should be considered. Clients in vulnerable roles (PCs that are used for outside communication for example,

[17] See http://blogs.technet.com/b/msrc/archive/2013/02/12/baseball-bulletins-and-the-february-2013-release.aspx.

or represent a vulnerable part of the infrastructure) require more and swifter patching than machines that are used for tasks that do not involve confidential information. For each available patch, carefully check if it should be deployed to the whole network, only to clients in certain roles, or not at all. SMB networks are a little easier to maintain in this respect, because the number of roles is more limited. Still, a distinction should be made between the different types of client roles in the network. Rolling out the same patch configuration to all machines is usually not recommended.

At this stage, it is also important to find out if the selected patches have any dependencies. Some software provides simple updaters that update any version of the software to the latest version. Others use incremental updates, which require you to backtrack across versions to find all appropriate patches, to be deployed in order. A software patch can also require a certain version of a driver, framework or other product to be installed. Information about such dependencies is usually available in the patch description, available with the patch download or directly from the software vendor. Dependencies can become increasingly complicated when planning an initial deployment round for systems that have not been patched for a while. Windows updates and service packs often depend on each other and can require a lot of attention to deploy correctly. Both during the strategy phase and during the testing phase, special attention should be paid to see if any patch requires additional installations.

Deployment is a relatively straightforward process, but some questions should be answered during the planning phase. First and foremost, decide on a point in time at which the patches should be installed. For security patches, administrators may instinctively choose a "faster is better" approach. While swift deployment will indeed quickly fix the security hole, there are a few more factors at play. To prevent compatibility issues, patches need to be tested, which can take several days. The actual deployment itself also needs time. In only a few cases, clients should be forced to reboot after installing a patch. If the reboot is not forced, it might take a while before the patch actually comes into effect. Depending on the network size, staged distribution might be required, further extending the deployment timeframe. For mobile and VPN workers, deployment might need even longer, because they might not always be connected to the company network, and not all updates can be deployed at once. (For clients with limited bandwidth, it might even be useful to take file size into account when prioritizing patches.)

As a general guideline, making sure critical patches are deployed between 48 hours to one week after their release is recommended. Non-critical security patches or functional updates can be delayed. However, as mentioned, the planning strongly depends on network layout, patch strategy and severity. Even in patch management, security has to be balanced with usability and availability. For some systems, downtime might be so counterproductive that patch deployment has to be delayed. For others, forcing the user to reboot may be acceptable. Also keep in mind that it does not matter when the patch is deployed, but when it takes effect. Reboots, staged distribution, or other delays may push the effective patch implementation back, requiring a (temporary) workaround to be implemented. However, a workaround should always be an interim measure: security holes are ultimately only reliably fixed by the proper patch.

To help streamline decision making, it can be helpful to specify a timeline in advance, taking all of the network's properties into account, with exploit severity as the only variable. Defining a patch deployment compliance level quantifies this effort. It should be defined as a fixed percentage after the first few days, gradually climbing up. Attaining 100 percent patch compliance is nearly impossible, because some machines can be in use very rarely, or are currently undergoing maintenance.

3.4 Step IV: Testing

The testing procedure is the most vital step in preventing complications during and after patch deployment. Productivity can be limited severely if patch deployment renders clients (temporarily) unusable. Unfortunately, the speed at which critical security patches should be deployed can sometimes limit the extensiveness of patch tests. As with planning which patches to deploy, security and usability are not always compatible. Realize that not all patches will, or even have to, go through the same procedure. Small, simple patches do not cause problems with network load, deployment or compatibility like larger patches may, so a lighter testing regime is in order. At the same time, the reduced severity rating of minor patches allow them to be tested for a longer period as they do not need to be deployed very quickly. In any case, every patch should be tested using a baseline that ensures a problem-free deployment. As in other stages of patch management, SMB administrators have a slightly easier job here. With less scenarios to test, the whole procedure can be carried out a lot quicker, enabling swift patch deployment.

In order to perform patch testing, a testing environment has to be defined. Ideally, this environment features all possible client configurations that exist in the network. Errors and exceptions can only be located by emulating the deployment experience as closely as possible to how it will be carried out in the real network. This means setting up several physical client PCs, each with a representative configuration of one of the network client roles. Complications arise when planning patch tests for machines that have a central role in the network, such as servers. It is very hard to set up a proper testing environment that takes into account all aspects of the network configuration. One option is the deployment of a virtualized testing environment, with virtual server(s) and clients. While this does allow for software and configuration problems to be located, the physical aspect of patch deployment (e.g. bandwidth, free disk space) cannot be tested properly on a virtual machine. Alternatively, a non-vital part of the actual network can be designated as testing grounds. This allows an administrator to test patches in a very realistic environment. As long as patches are only test-deployed to non-critical machines, an actual network test can be very helpful. However, besides the obvious increased risk of complications, this approach blends the testing step with the deployment step. Keeping track of machines used for tests and separating them from the rest of the clients can become complicated very quickly.

Once an appropriate set of machines has been chosen as testing environment, the patches to be tested can be rolled out. Check the vendor's patch description to see if there are any known issues or other problems that explicitly need to be tested on network. The first hurdle is the type of installer that the vendor chose to use. A popular standard is Microsoft's Windows Installer (MSI). Since most if not all Windows machines have the necessary framework installed, many vendors develop their installation using this technology. Windows Installer allows for easy version management and unattended installations, important technologies to enable patch deployment. However, it is not a completely self-serviced solution. To allow for a setup procedure without any user interaction, tests should be run to see if the MSI process does not throw unexpected errors. Missing installation media or cache files are among the most common causes of an interrupted MSI installation or patching procedure and can severely disrupt a patch rollout over multiple network clients. Vendors can also choose to deliver software with their own installer. While many companies use standard solutions, which may in turn again be based on MSI, some proprietary installers can cause confusion. Poorly documented or even undocumented installer behavior and patching standards complicate patch deployment. Running this type of installer in a testing environment provides an opportunity to look at its behavior and tweak its parameters for mass deployment.

From a usability point of view, it is important to check if the patch installation requires a reboot. When files that need to be patched are in use during patch deployment, they can only be replaced once the client is restarted. It may be necessary to check which background services make use of files that need to be patched, and shut them down temporarily before initiating the patch procedure. For more complex patches, such as Windows security patches or service packs, a reboot is almost always required, regardless of prevention measures. If it turns out a reboot is required, the deployment should be carefully planned in the next step of the patching procedure (Schedule & assessment) to fall outside of or integrate with end user schedules or maintenance windows.

After finishing up the installation procedure, including one or more reboots, if necessary, the system should generally still be functioning. Except for changes made by the patch, software should still function as it did before. The system should still be able to boot, in a reasonable amount of time, and the end user should not be surprised by dialog windows, message boxes, cleanup processes or other remnants of patch installation.

Not just the installation procedure should be tested. Future compatibility problems can require a patch to be uninstalled, or users encounter problems in previously untested scenarios. Ensure that the software can also be restored to its pre-patch state. For standardized (MSI) installers, this can be a very simple procedure. For software for which program files are not version managed, or products that use custom installers, it can be necessary to create backup files before deploying the patch, to allow for later patch rollback. Some patches are so pervasive that there will not be a possibility to uninstall them without redeploying a substantive part of the system (such as Windows service packs). Make sure to read all documentation that accompanies the patch, to be able to anticipate compatibility issues, installation complications or other problems. If there are any uncertainties which are not clarified during the testing procedure, either decide to move the patch to the next patch cycle, pending further testing, or deploy it in stages, carefully monitoring its effects on the first few deployment groups.

The patch testing procedure is not only meant to check out the deployment, but also to gauge the effects on end users. After successfully installing a patch, software may function differently than before, due to new functions being added or a mitigation strategy for a critical issue. The testing environment should feature end user work environments, to allow for a quick comparison of pre- and post-patching situations. Depending on the amount of time spent on testing, various end user workflows should be tested. A patch deployment does not need to be completely transparent, but changes should be documented in advance and communicated with the end user well before the actual deployment.

When the deployment process and patched software themselves function without problems, patch testing can focus on external factors. Patched software may not be recognized by enterprise white- or blacklisting software, especially if its executable files have been altered. Make sure to edit application control lists to feature the updated version, or use a different level of granularity: filter the software by product name, version or vendor. Similarly, Windows policies might interfere with some functions of the patched software. If the testing procedure uncovers any potential conflicts, try to reconfigure the software or adjust the problematic policy.

3.5 Step V: Schedule and assessment

Once the goal of the patch, its functioning and its dependencies have been laid out, the actual deployment can be scheduled. This step consolidates information from almost all of the previous tasks in the patch management procedure. Network inventory, patch dependencies and deployment behavior lead to a single deployment plan. This step of the patch management policy provides some opportunities to reuse or consolidates prior practice for enterprises that have adopted standards such as ISO 27002 and may already have implemented change management and risk assessment policies. At the very least, a procedure to handle deployment exceptions must be developed, as well as a formal risk assessment for each patch.

By now, it is known which patches will have to be rolled out, and which severity rating they have been assigned. However, severity is not the only factor in deciding when to roll out a patch. Some patches can have specific system requirements or compatibility issues that need to be mitigated before deployment can be commenced. Furthermore, not all clients may be able to be patched at the same time. Some might be in use, others may have very limited deployment windows scheduled. In any case, distributing the patches gradually is recommended, as opposed to serving all clients with the updates at the same time. Unexpected conflicts can be spotted before the whole network has had the patch applied; in extreme cases the deployment can even be halted. If the testing procedure indicated possible difficulties with a certain patch, schedule deployment for that specific patch only on a few clients at first, expanding its reach only when rollout for the first machines has been successful.

Based on inventory and network client roles, clients can be divided into several groups, organized by the time at which they will be patched. While, as a general rule, the most vulnerable machines with the security holes of the highest severity should be patched first, practical circumstances can dictate otherwise. Important is to prevent unnecessary delays in patch deployment. The actual rollout process should be initiated soon after the testing and assessment phases are completed.

Physical limits can play a role in scheduling patch deployment. Distributing patches over the network can put a substantial strain on the infrastructure. If bandwidth is limited, patch deployment may have to be planned outside working hours. For clients that have only limited free space available, hard disk cleanup tasks need to be planned before deploying any patches.

Once a schedule has been worked out, clients can be notified of the scheduled deployment. Especially if the deployment phase will lead to a forced reboot or other types of client outage, letting end users know in advance helps to foster understanding. In the actual deployment phase, administrators can even choose to let users delay patch installations or reboots, if the client PC is needed for work at that time.

Patching production servers should be done very carefully. To make sure that reboots and unexpected downtime do not cause too much inconvenience, reserve a timeslot in advance and announce the maintenance to all users concerned. A fallback server should be available to prevent service disruptions if complications arise during the patching process.

3.6 Step VI: Patch deployment

At deployment time, all previous steps come together. With the patching schedule in hand, the physical task of distributing patches to all network clients starts. However, deployment is more than just pushing a patch installer to the clients. For older systems, for example, it can be useful to first force a full virus scan, especially if the system has not been patched recently. Lingering malware will be scooped up by the virus scan, to allow for a smooth patch installation.

After deployment, verification and reporting will help assess the effect of the patch deployment and help discover issues. Before anything is deployed to the clients, it should therefore be defined which actions will be logged or reported. Theoretically, any action that affects the client system should be logged to enable later analysis. Keeping tabs of all file system actions and registry mutations is one possibility, but be careful not to gather too much data. Never-ending activity logs hinder swift analysis.

Some vendors release diff-patches, publishing only the changes between one version and the next. While this saves a significant amount of disk space and bandwidth, an individual patch has to be released for each target version. To save time in building patches, most products are served with full patch files, which contain complete versions of the files to be patched. This simplifies cases where outdated clients would have had to be patched using multiple files, but the average patch file size is significantly increased. The patch deployment phase can therefore severely influence network bandwidth. All patches are downloaded once by the central patch management server, which subsequently distributes them to all applicable network clients. To prevent the patching process from disturbing other network jobs, try to limit the simultaneous network load. Deployment can be planned at a time when the network is not being used as heavily. Additionally, deployment should be staged. Not all clients need to be patched at the same time, for compatibility and performance reasons. Group clients together that are physically close to focus network usage only on certain nodes, or add a few clients from each network zone to a group to spread the load evenly. Even if the testing phase was completed without major exceptions or incidents, it is still recommended to group clients based on risk: deploy patches first to standard desktop clients or single platform server groups, before moving on to the more complicated machines in the network.

Patches do not necessarily need to be installed directly after they have been pushed to the client. Individual circumstances can delay patch installation. If patches are being pushed during working hours, when the clients are in use, usability is better served by delaying patch installation until shutdown, reboot or lock actions occur. Alternatively, configure the patch management software to push patch files only once the actual patching process is starting, combining both actions and allowing the complete process to take places during a scheduled maintenance window. Similarly, if reboots are required, these should be planned to minimize inconvenience. Patch installation and reboots can be forced, if the security risk outweighs the usability argument, but this should not be the default. Alternatively, users can be given the option to delay patch installation or reboots for a certain time, to give them the chance to finish their current work. To prevent software deployments from becoming out of sync with the rest of the network clients, the delays should be limited (e.g. to maximum one day for patch installation or one hour for reboots).

Before running and installing a patch file, its authenticity should be verified. At several stages, files may become corrupted, intentionally or not. Vendor servers are trustworthy, but patch files can still get damaged during the initial download. To check the authenticity of patch downloads vendors often provide checksum files or hashes against which downloads can be checked. After further deployment in the network, it is again recommended to check patch files against their hashes to prevent unexpected results due to file corruption. For extra security, patch distribution over the network can be encrypted, but the performance loss might not be worth the increased reliability.

It is recommended to actively monitor patch deployment. If a patch fails to install, it can have consequences for other patch installations that depend on it. Lack of free disk space is one very common reason for patch installations to fail, but there can be many others. Patch installers often return error codes when something goes wrong while running in unattended mode. Manual intervention is often necessary when patch deployment goes wrong. For that reason, the patch management policy should define how to handle exceptions, and within which timeframe.

3.7 Step VII: Verification and reporting

Although verification should already take place during the deployment process, the moment patches have been deployed to all network clients marks the beginning of the formal verification and reporting process. The post-deployment review consists of analyses of deployment logs and exceptions, as well as formal checks to see if all planned deployments have been carried out correctly. If any exceptions occurred during the patching process, analyze what happened exactly. By identifying the cause of an installation issue, further patch deployment can be streamlined. Some errors may occur because of individual problems with client hardware; others can be tracked to general domain policy or network load issues.

Aside from examining logs that have been created during deployment, the installation result can be checked in several ways. As part of the patch deployment policy, a complete inventory of software versions on each network client should be available. Refreshing the inventory and comparing it against the previous version provides an accurate way to check if software has indeed been updated to the desired version. However, not all patches are set to increase the software version number. Minor patches, patches without a proper installer, or even vendor oversight can lead to erratic version numbering, or none at all. Listing file checksums can help, but increases the amount of data to be reported, which can be overwhelming. If enough documentation is available, technical details about the vulnerability can reveal an easy way to check if the patch has been installed correctly: try to carry out the exploit to see if the issue has been remediated. Specialized vulnerability scans can perform automated network scans to see if any vulnerabilities still exist.

If a patch has not been installed correctly, there are several remedies. If network load or free disk space were the problem, reducing the load or freeing space will allow the patch deployment to be carried out successfully after a second try. However, if there are compatibility issues with existing software, or other exceptions, a manual installation might be necessary. Having remote access to the client in question is practical, to try to run the patch installer manually.

A non-automated but valuable way of verifying proper patch deployment is involving the end user. Even though the actual patch files might have been installed properly, compatibility problems may occur when other software depends on an exact version of the patched product. Moreover, it is easier to spot performance problems in the daily workflow of the end user, than in an artificial testing environment. Especially for clients with non-standard deployments, the risk of problems is larger. By giving users the possibility to provide feedback about patches, administrators are made aware of problems they could not find out by themselves. End user feedback will often lack the specifics necessary to pinpoint the problem immediately, but can be a great indicator of patch performance.

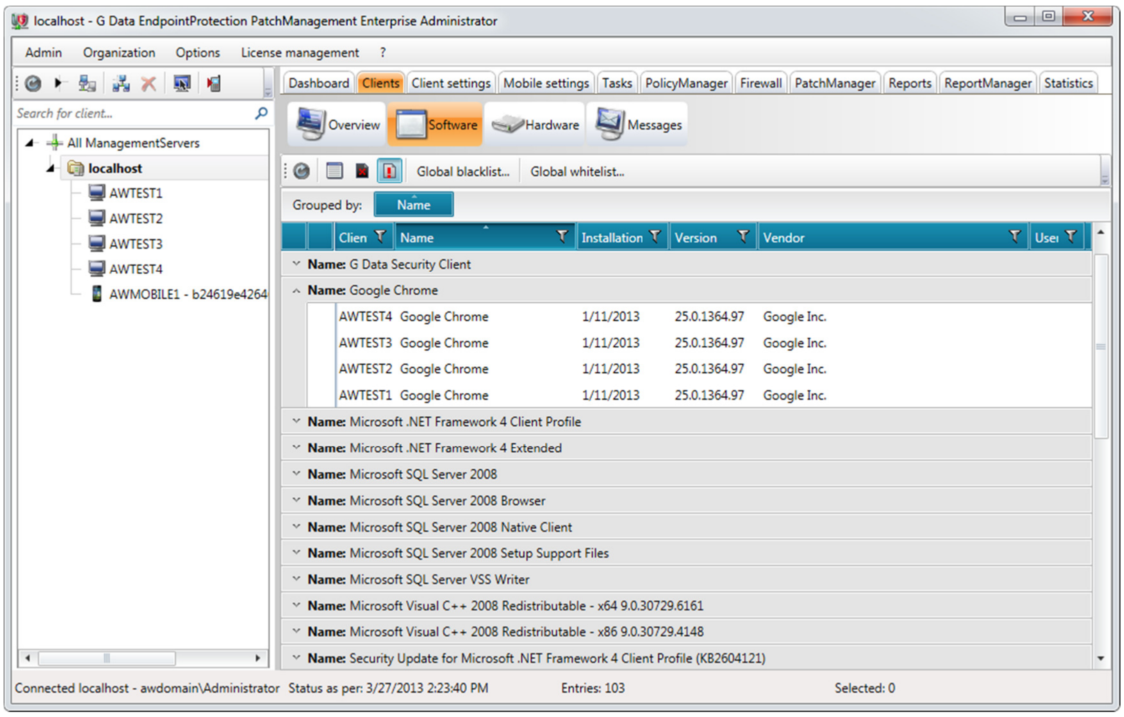
With or without user feedback, there should always be the technical possibility to roll back patches. The testing phase should have guaranteed that the patches that have been deployed do not cause major problems, but in individual cases, performance or functionality loss can always occur. The administrator should be able to centrally initiate a patch rollback procedure.

4 Automating patch management with G Data Patch-Manager

G Data offers patch management as part of its business solutions. The functionality is available as an optional module for all users of Business and Enterprise products. Combined with the existing modules of G Data AntiVirus, ClientSecurity and EndPointProtection, functional support for all steps of the patch management procedure is provided.

4.1 Step I: Inventory update

Firstly, it is important to take and keep an inventory of machines in the network and their software and hardware. To support the patch management procedure, it should be known which software versions are in use on the company network at any time. G Data provides a streamlined inventory tool as part of its business solutions. The Clients module lets administrators access a full list of installed software for each network client. The inventory can be organized to provide different types of information. The default view shows a flat list of all software that is installed on the selected clients. The listing includes the installation date, the software vendor and the currently installed version. By grouping the items according to name, for each product a quick overview is available to check if the latest version has been installed on all machines.


localhost - G Data EndpointProtection PatchManagement Enterprise Administrator

Admin | Organization | Options | License management | ?

Dashboard | Clients | Client settings | Mobile settings | Tasks | PolicyManager | Firewall | PatchManager | Reports | ReportManager | Statistics

Overview | Software | Hardware | Messages

Global blacklist... | Global whitelist...

Search for client...

- All ManagementServers
  - localhost
    - AWTEST1
    - AWTEST2
    - AWTEST3
    - AWTEST4
    - AWMOBILE1 - b24619e4264

Grouped by: Name

Clien	Name	Installation	Version	Vendor	User
Name: G Data Security Client					
Name: Google Chrome					
AWTEST4	Google Chrome	1/11/2013	25.0.1364.97	Google Inc.	
AWTEST3	Google Chrome	1/11/2013	25.0.1364.97	Google Inc.	
AWTEST2	Google Chrome	1/11/2013	25.0.1364.97	Google Inc.	
AWTEST1	Google Chrome	1/11/2013	25.0.1364.97	Google Inc.	
Name: Microsoft .NET Framework 4 Client Profile					
Name: Microsoft .NET Framework 4 Extended					
Name: Microsoft SQL Server 2008					
Name: Microsoft SQL Server 2008 Browser					
Name: Microsoft SQL Server 2008 Native Client					
Name: Microsoft SQL Server 2008 Setup Support Files					
Name: Microsoft SQL Server VSS Writer					
Name: Microsoft Visual C++ 2008 Redistributable - x64 9.0.30729.6161					
Name: Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.4148					
Name: Security Update for Microsoft .NET Framework 4 Client Profile (KB2604121)					

Connected localhost - awdomain\Administrator | Status as per: 3/27/2013 2:23:40 PM | Entries: 103 | Selected: 0

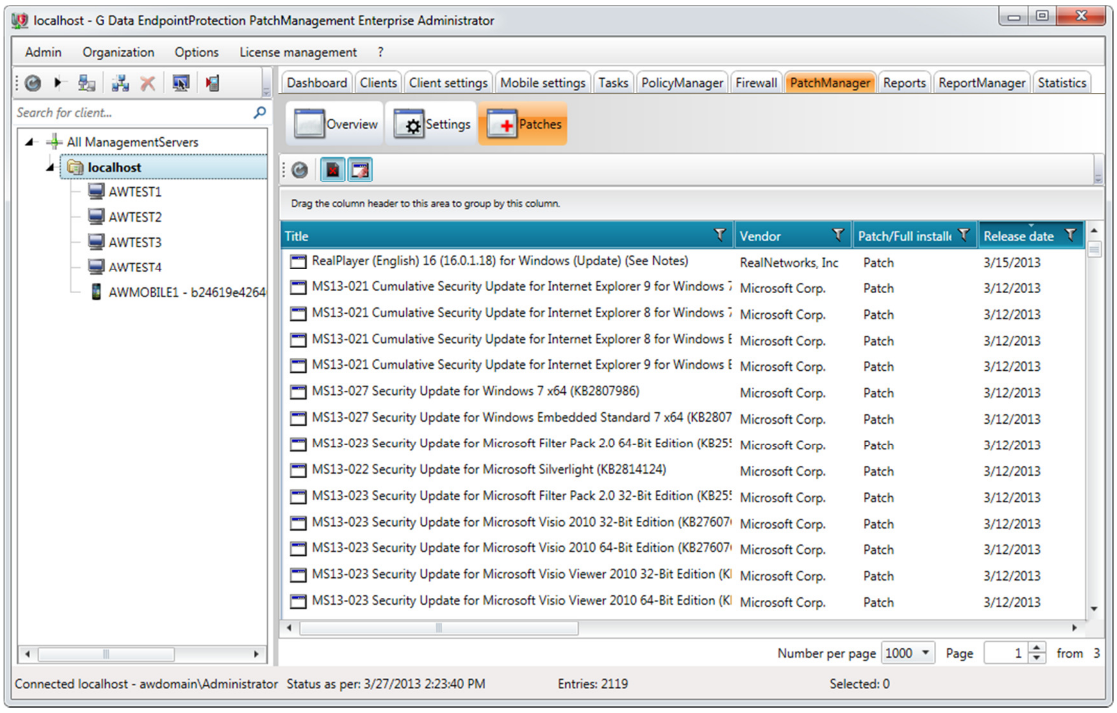
***Image 7**: G Data Administrator, Client Software Inventory view*

At this stage, it is recommended to check if network clients are running any software that is not part of the standard deployment. Administrators cannot be aware of potential security risks for all software in existence. Using a software inventory helps spot unsanctioned software installations. Administrators can decide to either add the software to their official deployment list (Whitelist), or to remove them and block them from being installed or run (Blacklist). Users of G Data EndpointProtection can use the PolicyManager module to apply network-wide policies, whitelisting or blacklisting software to control deployment.

Not only is it important to keep track of software; successful deployment also depends on physical prerequisites like network load and hardware specifications. The latter can be listed using the Hardware inventory function. A wide range of specifications can be tracked. Physical specs, like CPU speed and the amount of internal memory, help predict patch deployment speed and performance. Important is knowing the amount of free disk space available, to prevent patch deployment from generating errors. Additionally, bios and motherboard firmware versions can be tracked, to compare against newly published firmware.

4.2 Step II: Information gathering

As soon as an inventory has been established, administrators should keep up with information about the latest patches to compare to their existing stock. G Data's PatchManager module provides a list of latest available patches for a wide range of products on the Patches tab. The database is updated automatically as soon as vendors publish a new patch. Sort the list by Release date to see which patches have been recently released. More information, and often full release notes, can be obtained by right-clicking on a patch and checking its properties.



Title	Vendor	Patch/Full install	Release date
RealPlayer (English) 16 (16.0.1.18) for Windows (Update) (See Notes)	RealNetworks, Inc	Patch	3/15/2013
MS13-021 Cumulative Security Update for Internet Explorer 9 for Windows 7	Microsoft Corp.	Patch	3/12/2013
MS13-021 Cumulative Security Update for Internet Explorer 8 for Windows 7	Microsoft Corp.	Patch	3/12/2013
MS13-021 Cumulative Security Update for Internet Explorer 8 for Windows E	Microsoft Corp.	Patch	3/12/2013
MS13-021 Cumulative Security Update for Internet Explorer 9 for Windows E	Microsoft Corp.	Patch	3/12/2013
MS13-027 Security Update for Windows 7 x64 (KB2807986)	Microsoft Corp.	Patch	3/12/2013
MS13-027 Security Update for Windows Embedded Standard 7 x64 (KB2807	Microsoft Corp.	Patch	3/12/2013
MS13-023 Security Update for Microsoft Filter Pack 2.0 64-Bit Edition (KB25	Microsoft Corp.	Patch	3/12/2013
MS13-022 Security Update for Microsoft Silverlight (KB2814124)	Microsoft Corp.	Patch	3/12/2013
MS13-023 Security Update for Microsoft Filter Pack 2.0 32-Bit Edition (KB25	Microsoft Corp.	Patch	3/12/2013
MS13-023 Security Update for Microsoft Visio 2010 32-Bit Edition (KB27607	Microsoft Corp.	Patch	3/12/2013
MS13-023 Security Update for Microsoft Visio 2010 64-Bit Edition (KB27607	Microsoft Corp.	Patch	3/12/2013
MS13-023 Security Update for Microsoft Visio Viewer 2010 32-Bit Edition (K	Microsoft Corp.	Patch	3/12/2013
MS13-023 Security Update for Microsoft Visio Viewer 2010 64-Bit Edition (K	Microsoft Corp.	Patch	3/12/2013

Number per page 1000 Page 1 from 3

Connected localhost - awdomain\Administrator Status as per: 3/27/2013 2:23:40 PM Entries: 2119 Selected: 0

***Image 8**: G Data Administrator, PatchManager Patches view*

4.3 Step III: Strategy and planning

Administrators can directly check one or more patches against their client systems. To check a single patch for applicability, right-click it and click Check patches for usability. This schedules a Software recognition job for the specified client(s). Alternatively, an automatic scan can be carried out for each new patch that is added to the database. Using the Tasks module, plan a Software recognition job that is executed as soon as a new patch is available. PatchManager then checks the new patches for applicability across all specified clients.

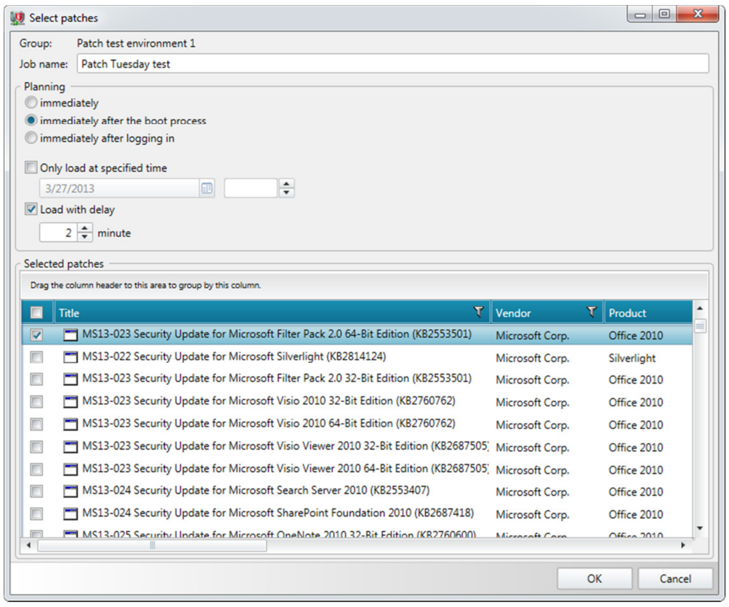
PatchManager does not automatically install patches, to allow administrators to select patches and plan patch tests beforehand. After scanning for applicability, select the appropriate server or client(s) in the client management area and open the Overview tab of PatchManager. Group the patches by dragging the Status column to the group bar above the list. This helps to quickly locate patches that are applicable, not applicable or have already been installed. Patches that are applicable for the client system(s) are the ones that need to be reviewed, tested and finally deployed.

To help decide whether to deploy a certain patch or not, PatchManager provides a set of information for each patch. In its list overview, the PatchManager module shows the products that a patch applies to, as well as its release date, its official title, and its priority. For each patch, a full description and usually a URL to the official release notes are provided. These pieces of information help administrators decide how severe a certain security issue is, and how quickly its patch needs to be deployed. Patches with a higher severity should be installed with a higher priority than non-critical patches. At the same time, not all software that is in use across the network is as important: critical applications should always be patched before less critical applications. This is where the patch management policy is relevant, by helping to decide quickly which patches to deploy in which order (see chapter 2.1).

The important point to remember is that not all patches should be installed by default. The point of patch management automation is not to take decision making out of the equation, but to provide enough details to make informed decisions, and to streamline the deployment process. PatchManager provides as much information as it can, but the decision to test and finally deploy a patch, is always up to the administrator.

4.4 Step IV: Testing

Once it has been decided that a specific patch will be deployed, the testing procedure can start (step 4). It is recommended to use a set of representative machines to test patches. These machines should be similar to the clients that are actually in use, in order to test for possible problems without disrupting the actual clients. However, not every administrator will have access to enough machines to build a small-sized replica of their network. Virtualization is the recommended method; if there is really no other solution, a non-vital subset of the network can be used. In any case, G Data can help manage the testing network. Using G Data Administrator, the test environment can be organized in one or more groups. Patches can be deployed to one or more clients in one or more groups, to observe the installation and its effects (see chapter 3.4 for more information about which machines to include in the testing environment).



Group: Patch test environment 1
Job name: Patch Tuesday test

Planning
- [ ] immediately
- [✓] immediately after the boot process
- [ ] immediately after logging in

- [ ] Only load at specified time
  3/27/2013
- [✓] Load with delay
  2 minute

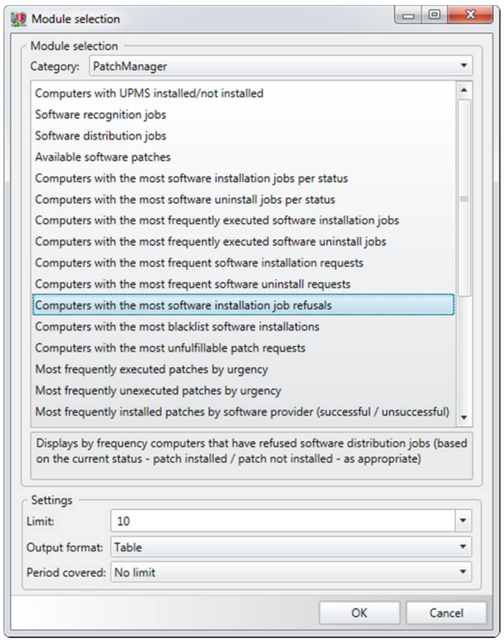
Selected patches

Drag the column header to this area to group by this column.

	Title	Vendor	Product
✓	MS13-023 Security Update for Microsoft Filter Pack 2.0 64-Bit Edition (KB2553501)	Microsoft Corp.	Office 2010
	MS13-022 Security Update for Microsoft Silverlight (KB2814124)	Microsoft Corp.	Silverlight
	MS13-023 Security Update for Microsoft Filter Pack 2.0 32-Bit Edition (KB2553501)	Microsoft Corp.	Office 2010
	MS13-023 Security Update for Microsoft Visio 2010 32-Bit Edition (KB2760762)	Microsoft Corp.	Office 2010
	MS13-023 Security Update for Microsoft Visio 2010 64-Bit Edition (KB2760762)	Microsoft Corp.	Office 2010
	MS13-023 Security Update for Microsoft Visio Viewer 2010 32-Bit Edition (KB2687505)	Microsoft Corp.	Office 2010
	MS13-023 Security Update for Microsoft Visio Viewer 2010 64-Bit Edition (KB2687505)	Microsoft Corp.	Office 2010
	MS13-024 Security Update for Microsoft Search Server 2010 (KB2553407)	Microsoft Corp.	Office 2010
	MS13-024 Security Update for Microsoft SharePoint Foundation 2010 (KB2687418)	Microsoft Corp.	Office 2010
	MS13-025 Security Update for Microsoft OneNote 2010 32-Bit Edition (KB2760600)	Microsoft Corp.	Office 2010

***Image 9**: G Data Administrator, Tasks, Software distribution job (Test)*

To deploy one or more patches to a test group, select the group in the client management area. Open the Tasks module and create a new Software distribution job. Select the patch(es) to be distributed and specify at which time this should occur. Selecting the patch can be made easier by grouping the patch list by Vendor or Product. Repeat this process with all appropriate patches and for all appropriate test groups. It is recommended to test only one patch per system at the same time, to be able to pinpoint possible problems on a specific patch.



Module selection
Category: PatchManager

- Computers with UPMS installed/not installed
- Software recognition jobs
- Software distribution jobs
- Available software patches
- Computers with the most software installation jobs per status
- Computers with the most software uninstall jobs per status
- Computers with the most frequently executed software installation jobs
- Computers with the most frequently executed software uninstall jobs
- Computers with the most frequent software installation requests
- Computers with the most frequent software uninstall requests
- Computers with the most software installation job refusals
- Computers with the most blacklist software installations
- Computers with the most unfulfillable patch requests
- Most frequently executed patches by urgency
- Most frequently unexecuted patches by urgency
- Most frequently installed patches by software provider (successful / unsuccessful)

Displays by frequency computers that have refused software distribution jobs (based on the current status - patch installed / patch not installed - as appropriate)

Settings
Limit: 10
Output format: Table
Period covered: No limit

***Image 10**: G Data Administrator, ReportManager module selection*

During the testing period, as well as the verification phase after deployment, the ReportManager module can assist in finding out what the status is of patches being deployed, and which machines are generating errors. ReportManager lets the administrator select several modules to be combined into one report. Its PatchManager module category provides several useful options, such as the patches most frequently not installed or computers with unexecuted software distribution jobs (which may point to installation problems), or computers with the most frequent patch requests or refusals (for analysis afterwards).

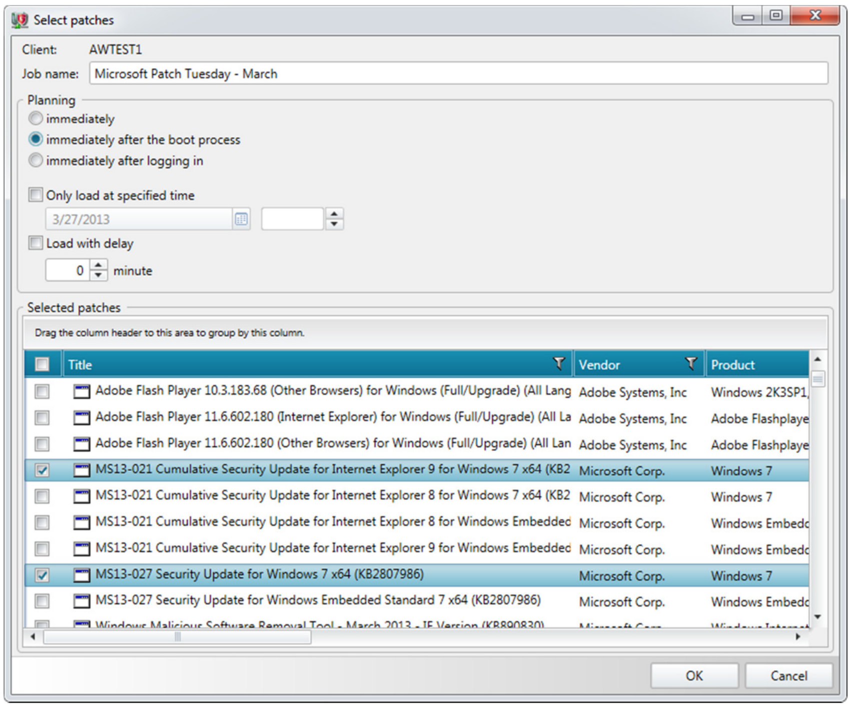
In addition to the ReportManager module, patch testing status can also be located in the Tasks module itself. Open the relevant task and check the details to see the status for each patch. If it appears that a patch has not been deployed successfully, update the Software inventory for that client to double-check. If the patch cannot be deployed, check the system locally and try a manual patch deployment. If a patch is causing problems during the testing phase, it should never be deployed automatically on a large scale.

4.5 Step V: Schedule and assessment

After finishing the testing stage, the actual deployment can be planned. With all applicable patches located and tested, a schedule can be set up. Using the patch management policy, decide in which order the patches should be deployed and to which (groups of) machines at first. Use the Messages function of the Clients module to notify clients of the patch schedule and to warn them of potential reboots.

4.6 Step VI: Patch deployment

For patches that have been properly tested, a Software distribution job can be planned. Use the Tasks module to schedule a Software distribution job with the appropriate patches for the appropriate clients. To prevent interference with end user workflows, patches can be scheduled to be deployed at a specific time, or directly after the next boot or login. An optional delay prevents patches from being deployed while other system-intensive processes may be running.


Select patches

Client: AWTEST1

Job name: Microsoft Patch Tuesday - March

Planning

( ) immediately
(●) immediately after the boot process
( ) immediately after logging in

☐ Only load at specified time

3/27/2013

☐ Load with delay

0 minute

Selected patches

Drag the column header to this area to group by this column.

☐	Title	Vendor	Product
☐	Adobe Flash Player 10.3.183.68 (Other Browsers) for Windows (Full/Upgrade) (All Lang	Adobe Systems, Inc	Windows 2K3SP1,
☐	Adobe Flash Player 11.6.602.180 (Internet Explorer) for Windows (Full/Upgrade) (All La	Adobe Systems, Inc	Adobe Flashplaye
☐	Adobe Flash Player 11.6.602.180 (Other Browsers) for Windows (Full/Upgrade) (All Lan	Adobe Systems, Inc	Adobe Flashplaye
✓	MS13-021 Cumulative Security Update for Internet Explorer 9 for Windows 7 x64 (KB2	Microsoft Corp.	Windows 7
☐	MS13-021 Cumulative Security Update for Internet Explorer 8 for Windows 7 x64 (KB2	Microsoft Corp.	Windows 7
☐	MS13-021 Cumulative Security Update for Internet Explorer 8 for Windows Embedded	Microsoft Corp.	Windows Embedc
☐	MS13-021 Cumulative Security Update for Internet Explorer 9 for Windows Embedded	Microsoft Corp.	Windows Embedc
✓	MS13-027 Security Update for Windows 7 x64 (KB2807986)	Microsoft Corp.	Windows 7
☐	MS13-027 Security Update for Windows Embedded Standard 7 x64 (KB2807986)	Microsoft Corp.	Windows Embedc
☐	Windows Malicious Software Removal Tool - March 2013 - IE Version (KB890830)	Microsoft Corp.	[illegible]

Image 11: *G Data Administrator, Tasks, Software distribution job (Deployment)*

4.7 Step VII: Verification and reporting

To verify and evaluate patch deployment, the inventory tools can be of great assistance. Additionally, G Data's PatchManager module offers the possibility for direct user feedback. If the administrator enables the respective option, end users can request patches to be rolled back, due to performance or compatibility issues. Patches that are applicable to the system, but have not been deployed yet or will not be deployed at all, can be requested by end users in case there is an urgent need to patch a product. The distribution and rollback request system integrates directly with the PatchManager module and allows the administrator to plan a distribution or rollback job directly from the Reports module. Consider the following example: an end user can no longer use application 1 and is waiting for a patch to be deployed. During the testing phase, the administrator discovers compatibility issues with application 2, and decides the patch will not be deployed to the network. The end user does not use the affected feature of application 2, and decides that the patch for application 1 should be deployed anyway. Through G Data's bidirectional patch distribution, the user can request the patch to be installed. Upon approval of the request, patch distribution will take place like it would have done in a normal deployment scenario.