PROCEEDINGS OF SCIENCE

A Race for Security: Identifying Vulnerabilities on 50 000 Hosts Faster than Attackers

Michal Procházka*
CESNET z.s.p.o., Zikova 4
160 00 Prague, Czech Republic
E-mail: michalp@ics.muni.cz

Daniel Kouřil
CESNET z.s.p.o., Zikova 4
160 00 Prague, Czech Republic
E-mail: kouril@ics.muni.cz

Romain Wartel
CERN, IT Department
CH-1211 Geneva 23, Switzerland
E-mail: Romain.Wartel@cern.ch

Christos Kanellopoulos
AUTH, Grid and HPC Center
Thessaloniki, Greece
E-mail: skanct@grid.auth.gr

Christos Triantafyllidis
AUTH, Grid and HPC Center
Thessaloniki, Greece
E-mail: ctria@grid.auth.gr

Unpatched security vulnerabilities are often misused by attackers to take over machines or cause other harm to computers and their legitimate users. Having proper and timely patch management is crucial to keep the system secure and resistant to common attacks targeting known weak spots. For this, a monitoring system that is able to provide a global view of the whole infrastructure is inevitable. In this paper we present service Pakiti that makes it possible to monitor patches across a number of machines and retain a fresh overview about the patching status. Using Pakiti a system administrator can detect machine where patching failed for whatever reason or was not initiated at all.

The International Symposium on Grids and Clouds and the Open Grid Forum - ISGC2011,
March 25-30, 2011
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC 2011 & OGF 31)031

1. Introduction

When operating a large scale grid infrastructure, it is important to manage a number of security risks that could impact its availability. The experience gathered over several years of operations in the EU EGEE infrastructure clearly shows that any infrastructure can be as weak as its weakest link. In particular, it is essential to ensure that all resources in the infrastructure offer a similar level of security.

Offering a homogeneous level of security across multiple, often heterogeneous, resources can be challenging. This is especially the case for applying software security updates, which can require service downtime, sufficient technical expertise and a sufficient degree of coordination. Unfortunately, failure to promptly apply security updates remains one of the main causes of security incidents affecting computing infrastructures, as we can conclude from the incidents observed in the Grid environment.

The experience indicates that a number of critical updates were not applied for the following reasons:

- Communication problems: faulty information flow, incorrect roles and responsibility definitions
- Infrastructure issues: only a part of the systems was upgraded, or not all the software update process was completed (e.g. no service/system restart)
- Insufficient staff expertise: a particular exploit did not work, staff concluded the site was secure
- Heterogeneous standards: the staff did not understand or agree with the implications of the risk
- Ineffective mitigation techniques: the infrastructure was modified to deflect the specific issue, but could still be attacked via a different vector
- Missing dependencies from hardware vendors: a proprietary kernel-level driver, critical for the system, has not been made available by the vendor, leaving the system unpatchable.

It appears that a number of technical, communication, or staff-related issues can therefore interfere with the security patching process, potentially creating a window of opportunity for attackers.

Pakiti has been designed to help managing these risks and offers a central view of the patching status of the infrastructure it monitors, based on its clients current state, and on the vulnerabilities. For example, it is possible to centrally display the exact list of hosts specifically vulnerable to a given vulnerability, based on its CVE number.

Keeping an overview is crucial not only for operators of large infrastructures like grid, but also for administrators and security staff of individual organizations whether or not they are connected to a more widespread environment. Pakiti does not depend on any Grid-specific features and therefore presents an attractive solution providing a primary and/or additional patch monitoring on site level, too.

In this paper we present a detailed description of the principles and architecture that Pakiti is based upon, as well as a few real-world examples demonstrating various modes of Pakiti utilization.

2. Vulnerability Management

Security attacks often target vulnerabilities in computer applications that provide the attacker access to the system. While some existing vulnerabilities are not published and silently exploited by attackers for breaches, other vulnerabilities are publicly disclosed and fixed by the product vendors. Each major software producer has a procedure ensuring a security vulnerability is corrected and the fix made available to the product users in a timely manner. The fixes are distributed as *patches*, which are published in a predefined way which is controlled by the vendor. After a patch has been released, the users are notified and advised to update their systems using the patch. Unfortunately, the current experience indicates that keeping systems up-to-date with regard to the latest security patches is quite challenging for several reasons and systems often fail to fix the vulnerabilities even if the patches do exist.

At the same time it is important to note that leaving a known vulnerability unpatched increases significantly the chance of being successfully attacked with a malicious code exploiting the vulnerability. For instance, once a serious security patch has been released, it is often subjected to a reverse engineering process in order for the attackers to obtain more details about the vulnerability and produce working exploits [1]. Several attempts have taken place in the past to automate the process of generating exploits based on publicly available patches [2]. Malicious code produced either by reverse engineering or based on the description of the vulnerabilities is used by attackers to penetrate the systems that have not applied the patch for whatever reasons.

The need for proper patching have been acknowledged many times in the past [3, 4, 5]. The importance of correct patch management is also demonstrated by several recent Internet worms and botnets (e.g., Conficker and Stuxnet) that often try to obtain non granted access via unpatched yet known and fixed vulnerabilities. The importance of patch management is inherently present in any contemporary system, regardless if it is an operating system or application. Despite if the vulnerability can lead to access to the system or gaining elevated privileges, the harm it may cause is usually very immense.

There are a lot vulnerabilities known and new ones are emerging every day as people publish them in one way or another. In order to ease the management of vulnerabilities, the *Common Vulnerabilities and Exposures* (CVE) list has been implemented [6]. With CVE, each new vulnerability is assigned a unique identifier, which is used whenever the vulnerability is discussed. For instance, security advisories issued by software version are usually CVE-compatible so it is easy to identify the vulnerabilities they refer to. The CVE dictionary is operated by Mitre, Inc. and is freely available on the web.

In order to facilitate the exchange of information about security vulnerabilities, the OVAL language [7] has been standardized. OVAL is an XML dialect allowing to specify a vulnerability and a complete set of conditions that identify it. The conditions describe at least the version of software components and also the specification of the operation system and its flavors.

OVAL is supported by some large OS vendors like RedHat, SuSE and Microsoft, that issue a new updated version on each release of security patches.

All contemporary operating systems and often complex applications, provide mechanisms for applying security patches from the vendors. A key issue that must be addressed in patch management is a question which particular patches should be actually applied in a given environment. The severity of vulnerabilities depends on the local risk analysis and requirements. Therefore, it is common practice for sites to leave some vulnerabilities unpatched, for instance if they cannot be exploited in a given environment due to mitigating precautions applied, or if the patch itself might pose a larger risk (e.g., if a patch makes a mission critical crash, it may more acceptable to accept the risk of leaving a vulnerability unpatched).

Therefore, it is utterly legitimate that a single patch is applied only on a subset of all the machine for which it is applicable. The decision about whether or not a patch should be installed must be taken by the administrators. Therefore updates cannot be installed automatically from the repositories and the process must be triggered by the administrator, which makes the process more fragile. Especially in a large deployment where machines go down unexpectedly or become unavailable for a while, it is difficult keep an overview about the current status and find out which machines should be updated with which patches.

One of the key role of Pakiti is to provide an independent monitoring, which provide a view of the infrastructure and can detect problems in the patching procedures and/or systems.

3. Pakiti

Pakiti was initially developed in 2004 at the Rutherford Appleton Laboratory in the UK, by Steve Traylen. It provided a simple client and server, enabling the team to detect the nodes where patches were missing. The tool was later taken over by Romain Wartel (Rutherford Appleton Laboratory / CERN) and was transformed into Pakiti, hosted on SourceForge[1]. It features support for multiple platforms, the distinction between security and standard patches, as well as a redesigned logic on the server side, enabling very lightweight clients. The tool has subsequently been adopted by the EU EGEE Operational Security Coordination Team, to monitor the security patching status of its 200+ sites, which comprise the European Grid Infrastructure.

Pakiti is currently used daily by many organizations and computing infrastructures across the globe, including the EGI CSIRT (successor of EU EGEE). While the project is hosted on SourceForge, the main developer, Michal Procházka, is also involved in the EGI CSIRT.

3.1 Basic design

Pakiti was designed as a client-server solution, with the clients gathering the list of all installed packages on a monitored host and sending them to the Pakiti server. The server processes the list trying to match the data with the current information about security patches. The results are made available either via a web-based GUI or remote API.

Typically, there is a single Pakiti server collecting data from an organization. The hostname of the Pakiti server is given in the client configuration and verified during the SSL/TLS handshake in order to reduce the risk of leaking potentially sensitive information. Also the transfer from the client is realized using HTTPS, which ensures sufficient confidentiality of the data. The machine

[1] http://pakiti.sourceforge.net/

running the Pakiti server should be also reasonably secured since its compromise may reveal weak spots in the infrastructure like machines that have not been patched and are vulnerable to specific exploits.

The client is a simple Bash script, which can run as a non privileged user. Based on the monitored host system, the client runs common distribution binaries like *dpkg-query* or *rpm -qa* to get the list of installed packages. Additional information like operating system version and release, kernel version and hostname are also gathered. All the information is sent to the Pakiti server using HTTP POST over HTTPS, the Pakiti client can use the host certificate to achieve mutual authentication of the server. No complex logic or data processing is done on the client.

The clients can be run and operated in several ways, depending on the preferences of the site administrators. The most common possibilities are jobs periodically triggered by cron or obtaining data using probes launched by a site monitoring (e.g., Nagios services).

The Pakiti server can process the reports either synchronously or asynchronously, based on the configuration directives. Acting in the synchronous mode, the Pakiti server processes data from the client immediately and the results are sent back to the client that is waiting for the response. Right after the data is sent to the server, the client will optionally receive a list of packages that should be upgraded. Packages in the list are tagged according to the type of the update (bugfix or security). On the other hand, when using the asynchronous mode, the client does not wait for any results and just feeds the server with the data and closes the connection. Following that, the server processes the data of all hosts on a regular basis, usually once a day. The asynchronous mode is preferred in large scale deployments with thousands of monitored hosts, where the clients reports are not uniformly distributed in time and thus are causing peeks overloading the server.

The Pakiti client can be configured using several configuration directives placed in a file or directly in the bash script itself.

Pakiti provides a simple web-based interface to access the results and work with the information collected. The access to the pages is limited using access control lists specified in the configuration. As described later, Pakiti is able to take authorization information from other services in the infrastructure, which eases its integration into existing environments. Using the web interface one can also configure the Pakiti server with settings needed to process the data received from the monitored machines. The Pakiti server stores all information in a relation database, currently using MySQL in the non-transactional mode. The server is coded in PHP and running as a web application in the Apache HTTP server.

The server is divided into two parts: the data processor and presentation layer. The data processor collects the data sent by the client and process it. In particular, upon receiving the report, the whole list of the packages is stored in the database and then each package version is compared to the versions from the vendor's repositories and CVEs.

The presentation layer is realized by a simple web GUI providing several views on the data stored in the Pakiti database. The GUI provides a list of domains and hosts together with basic statistics showing number of packages which are not up-to-date and which have security updates available. Hosts can also be organized into the tags (organization domains), the tag is specified in the Pakiti client configuration. A detailed view on every host is also available. The GUI also support searching. The hosts can be search by installed packages or CVEs to which the host is vulnerable.

The Pakiti administrator can specify which vendor's package repositories will be observed for up-to-date package versions. Each package repository is assigned to the operating system group it represents. As Pakiti checks preconfigured repositories, which should be used also on monitored hosts, it can provide the information about misconfigured hosts. Pakiti currently supports only RedHat OVAL definitions because other vendors of interest (e.g., SuSE) do not follow exactly the OVAL standard, which makes the OVAL processing impossible at the moment.

The package/host is considered vulnerable if a package version is lower than one in a package repository containing security updates assigned to the host's operating system or the package with the current version is listed in some CVE definition. CVEs in the RedHat OVAL definitions are categorized by the severity, therefore Pakiti labels CVEs using different colors. We found, however, the severity ranking used by RedHat to be focused mainly towards desktops systems and thus unsuitable for our purposes. For example, escalations to root privileges in Linux kernels are often not considered critical, while rather minor browser vulnerabilities score much higher. Therefore we have added CVE tagging, which allows an administrator to tag a concrete CVE. An administrator can define any number of tags and assign them to CVEs. Hosts found to have packages related to a marked CVE can be viewed on a single page in the GUI. Using the tagging mechanism we can easily detect machines exposing vulnerabilities deemed critical and need not rely on severity score assigned by the operating system vendor.

Pakiti is very sensitive about packages compiled manually and installed from a package, because they cannot be checke with any repository or OVAL. If a package is vulnerable and local administrator locally fix the vulnerability, he/she usually adds some additional string to the version of the package leaving the original version number to distinguish the package form the vendor's one. Pakiti then checks the version of the package, which does not change except the end string and marks the package as vulnerable, which leads to a false positive. Therefore, an exception can be added for each CVE and package version. Pakiti is able to show all package versions installed on monitored hosts according to the particular CVE, making it easy to select versions that should not show up in the reports.

Pakiti supports access control to its GUI. There are three roles recognized when making access control decisions: Pakiti administrator, Pakiti viewer and anonymous viewer. A Pakiti administrator can view all the results, manage ACLs and configure the repositories and OVAL sources. A Pakiti viewer can see the results, but only the one concerning his/her site/domain as specified by the in the corresponding ACL. An Anonymous viewer can only see results defined by an anonymous link (see below).

3.2 Additional features

Pakiti provides several additional features, which try to cover the needs arose during deployments in different organizations.

When showing a package or CVE for a particular site/domain, Pakiti can show anonymous links, which can be used by anonymous (unauthenticated) users to see these results. In order not to reveal data forever, the links have a limited lifetime. The anonymous links have all their parameters hashed together with a secret defined in the server configuration file, which makes it impossible for an anonymous user to spoof the link parameters. This feature is useful in situations where a

site administrator wants to engage other persons to help with patching the hosts, but do not want to reveal all the information about the site and provide a full access to the site results.

Pakiti is modular enough to be integrated into an existing monitoring infrastructure. Data from the client can be transferred to the Pakiti server in two ways. The client can either send the data directly to the server using HTTPS as usual, or it can print the data to its standard output, to let another monitoring tool transfer the data to the server using another messaging mechanism. In the later case the client can be run by monitoring tool itself. We have a particular experience with an integration of Pakiti into an Nagios-based monitoring infrastructure. Nagios probes launch the Pakiti client as an active check and make sure the results are delivered back to the Nagios server, instead of Pakiti. Having collected the results, the Nagios server passes on the data to the Pakiti server using a Pakiti proxy client located on the Nagios box. Proxy client only accepts data from the standard input and prints results of the check to the standard output. The Pakiti server checks whether the host running the proxy is authorized to send the results on behalf of other hosts. Using this approach the Nagios box have a complete knowledge about the patching status of the monitored nodes.

Pakiti can be also integrated with an existing authentication and authorization services. For authentication, Pakiti fully relies on Apache and takes over the identity authenticated by the Apache mechanisms. Authorization is also based on the same identifiers. The authorization rules are defined as pairs of identifiers and corresponding site/domain names to which the identifier has a read permission. Therefore ACL can be make use of any external authentication/authorization system, like any roles defined in site LDAP or grid-level GOC DB,

Pakiti server requires access to the Internet to get the up-to-date list of packages and their versions from the repositories and also to get the current OVAL definitions. If the Pakiti server is located behind an HTTP proxy, it can be configured to use the proxy for all outgoing connections.

4. Real-world Deployments of Pakiti

Over the past few years Pakiti has been deployed in many environments, often providing a key component of the security infrastructure. This section provides more details about particular utilizations of Pakiti in various environments, mostly related to Grids.

4.1 Pakiti in EGI

Pakiti has proven extremely useful for the European Grid Infrastructure (EGI) and its preceding infrastructures, mainly those operated by the the EU EGEE line of projects.

The Pakiti probes have been integrated with the project-wide monitoring services, which enable us to monitor all the sites connected to the Grid. To launch a probe on a site we do not require any special privileges granted by the site administrators and we re-use entirely the established monitoring infrastructure. The monitoring probes are submitted against the sites as common jobs and collect data from the worker nodes they land on. Using this approach we obviously can only access the compute facilities of sites and not other kind of services like storages or core services. On the other hand, this way of accessing worker nodes follows the way how the services are used by the users and potential attackers coming through the Grid infrastructure and therefore, it detects the most critical parts exposed by the sites.

A first Pakiti instance was set up in late 2008 and started collecting data from all the EGEE sites. During several months in 2008/2009 the code got stabilized and was made scalable enough to cope with the number of machines monitored. It also became the only mechanism to find out patching status of the site. The first results revealed some sites not applying security patches at all or incorrectly.

A slight drawback of using the job submission framework is that there is no guarantee about the actual worker node where the probe will land on, since it is completely dependent on the configuration of the local batch system. Due to this limitation we cannot monitor every single node on the sites nor rely on being able to produce statistics for a particular one. Instead, we have to cope with rather random selection of nodes. In practice the EGI instance of Pakiti purges the data on a daily basis to make sure old records with no Pakiti updates vanish from the views and do not distort the results.

In order to provide access to the results of a particular site only to its administrators, Pakiti synchronizes its access control lists with the EGI Grid Operation Center Database (GOCDB). Site administrators can utilize all the reporting and searching functions of Pakiti, but they cannot see the results of other sites or change any configuration options. Moreover, the EGI Pakiti instance provides anonymous links, using which other persons involved in the administration can see the results for pre-defined query.

Currently, EGI Pakiti monitors around 1600 hosts from 309 sites with 865, on average, installed packages on the monitored host. One report takes about 1 second to be processed, because the Pakiti clients are executed sequentially by the EGI Nagios. During the EGEE project, the tests were run in batches, making the Pakiti serve a large number of hosts at one moment, which increased the processing time of a single report to tens of seconds. Current EGI Pakiti server is divided into two virtual machines, a web server and database server, which both have dedicated 4 CPUs and 2 GB RAM. In this configuration, the load of the servers is unmeasurable.

The EGI CSIRT team uses Pakiti always when a vulnerability appears, which is internally ranked high for the Grid. Pakiti then serves to follow up with sites that expose worker nodes where the patches have not applied or contain other potentially vulnerable packages. During the final stage of the EGEE there were two incidents regarding a vulnerable Linux kernel. The EGEE security team sent a broadcast to all connected sites demanding to perform update. After 14 days, only a couple of sites had the kernels upgraded, as detected in Pakiti. This progress was unacceptable, therefore the security team established a new procedure defining a seven day period during which a critical vulnerabilities must be fixed. Sites failing to apply the updates within the time-frame may be suspended by the EGI CSIRT and thus disconnected from the infrastructure. These rules were clearly communicated to the sites along with advises how to make sure patches are applied correctly. When a similar situation emerged later, it took only 14 days to have all worker nodes upgraded.

Pakiti also helped in situations where administrators managed to upgrade the kernels, but a fraction of the machines has booted an older one, for whatever reason. Recently, the EGI CSIRT team had to handle several incidents regarding a security hole in the linux kernel and libc libraries. The majority of worker nodes had been fully updated within the 7 day period, while the rest were handled separately, mainly because of nonstandard configurations or problems in the communication channels. The overall progress since starting to use the Pakiti is clearly significant. Pakiti

allowed us to move from an infrastructure without any global control, where quite a lot of sites did not update regularly their systems, to an infrastructure, where worker nodes are patched to all known exploitable vulnerabilities.

4.2 Other deployments

As the Pakiti is freely available from SourceForge.net and is covered by an open-source license, it makes an attractive choice for other institutions. We are aware of several Pakiti deployments, both testing and production and are in touch with the administrators who share with us their experience and requirements on Pakiti. The following paragraphs provide a basic overview of these deployment demonstrating its broad utilization.

There is a deployment of Pakiti in the US Open Science Grid (OSG), where Pakiti monitors hosts at ten Tier-3 sites. The Czech National Grid Infrastructure (MetaCentrum) uses Pakiti to monitor all its hosts, both worker nodes and service hosts, with currently more then 950 machines being monitored. The Canadian Research Agency (VPAC) monitors around 100 heterogeneous hosts and integrated the Pakiti client into their configuration automatically managed by Puppet. German GridKa monitors about 50 machines. However small this number may seem, it actually hides a much richer infrastructure behind, since these machines present only specimens, each one representing a large group of equal installations. They monitor login machines, vo-boxes, and administrative servers using Pakiti. The Science & Technology Facilities Council (UK) monitors 830 machines and is going to scale up to 1200. Another deployment examples pose French sites GRIF (400 hosts) and LAL (70 hosts), where the Pakiti client configuration and part of the server configuration is managed by Quattor. Pakiti is also employed to monitor cca one thousand nodes comprising the post-South-East Grid federation of the EU EGEE project (consisting of Greek, Serbian and Cyprus clients). The AUTh university in Thessaloniki uses Pakiti to monitor about 200 nodes of different types, ranging from common services (like web and mail servers) to Grid-specific nodes.

5. Related Work

Pakiti belongs among services to ease the management of large-scale deployments. While bootstrapping and centralized configuration and installation of nodes is supported by a number of services (e.g., Puppet, Quattor), we are not aware of any system focusing on patch monitoring specifically. Updates can be applied using the tools provided by the operating systems and/or applications, but these tools only address the actual updating phase and do not provide an aggregated view on the whole system or security-specific features, like searching for particular CVE. Pakiti resembles commercial patch management services, which are provided by several commercial vendors. These tools, however, often tend to be all-or-nothing solutions, which are difficult to integrate with an existing infrastructure or adapt to non-standard requirements. Pakiti, on the other hand, provide only monitoring of the patches. It is not meant to actually apply the patches, which enables the system to remain a simple and light-weight enough to be a compelling alternative and/or addition to other management tools.

6. Conclusion

In the paper we presented service Pakiti to centralized patch monitoring. The service has been improved a lot during several past years yielding a mechanism, which can be reliably used to monitor deployments consisting of several thousands machines. We provided concrete examples of how Pakiti is used in contemporary Grids and other large-scale infrastructures. The Pakiti service has also played a crucial role in improving responsibility of the Grid site of the EU EGEE project, so as in raising awareness about severity of critical security bugs. The modularity of the Pakiti architecture and openness of the sources make it possible to utilize Pakiti in a number of scenarios and deployments.

Even though the paper focused mainly on Grids, Pakiti is open enough to be used in a number of other infrastructures as well. We are ready to support user from other areas as well.

Acknowledgment

The work on Pakiti has been supported by the following projects, whose support is highly appreciated. The EU "EGI-InSPIRE project" (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe), contract number: RI-261323, research intent "Optical Network of National Research and Its New Applications" (MSM 6383917201) of the Ministry of Education of the Czech Republic.

References

[1] J. Oh, "ExploitSpotting: Locating Vulnerabilities Out Of Vendor Patches Automatically," in *BlackHat 2010*, July 2010.

[2] D. Brumley, P. Poosankam, D. Song, and J. Zheng, "Automatic patch-based exploit generation is possible: Techniques and implications," in *2008 IEEE Symposium on Security and Privacy,*, April 2008.

[3] SANS Institute, "The SANS Top 20 Internet Security Vulnerabilities," http://www.sans.org/top20/2004/, 2005.

[4] "Symantec Global Internet Security Threat Report trends for 2009," http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_% internet_security_threat_report_xv_04-2010.en-us.pdf, April 2010.

[5] P. Mell, T. Bergeron, and D. Henning, "Creating a Patch and Vulnerability Management Program," http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf, November 2005, recommendations of the National Institute of Standards and Technology.

[6] The MITRE Corporation, "CVE – Common Vulnerabilities and Exposures (CVE) home page," http://cve.mitre.org/.

[7] The MITRE Corporation, "Open Vulnerability and Assessment Language – Element Dictionary," http://oval.mitre.org/language/version5.8/ovaldefinition/documentation/oval-definitions-schema.html.