CANONICAL

Ensure security and uptime when patching vulnerabilities in Linux

Considerations for scheduled and unscheduled maintenance

October 2022

Executive Summary

Security vulnerabilities pose a threat to your production environments in two ways. First, there is the risk a malicious actor could turn the security vulnerability into an exploit. Second, there is the risk that your attempts to mitigate vulnerabilities have unintended side effects. For example, you may face business disruptions due to changes in the software, staffing constraints and human errors.

To avoid these disruptions and ensure uptime while performing security patches, organisations should have sensible configurations for software that can be patched through multiple channels. In addition, they should have protocols in place to inform timing, prioritisation, and testing.

In this paper, we will establish the baseline for sufficient risk mitigation when applying security patches. Our approach will cover best practices for on demand vulnerability scanning and on demand security patching outside of the normal maintenance windows. Next, we will cover various scheduled security patching approaches. And lastly, we will review mechanisms to roll out security patching in an automated manner as the patches become available.

About Canonical and Ubuntu

Ubuntu is available as a fixed release Linux distribution, and Canonical is responsible for its six-monthly milestone releases and the regular long term support (LTS) releases. Canonical offers a range of services and products for organisations, enterprises, and individuals, to help them secure and manage their Ubuntu estate. Ubuntu LTS comes with a security maintenance commitment of 10 years with an Ubuntu Pro subscription. This also covers security patches for over 25,000 open-source software packages in Ubuntu's repositories (currently available in beta). These packages include popular applications and toolchains like Python 2, Puppet, phpMyAdmin, Ansible, Redis, Rust, Drupal, Zookeeper, Nagios, Node.js, Tomcat and many more. All of these software packages can be updated from within Landscape, which provides system administration at scale for Ubuntu anywhere: on public clouds, private clouds, at the edge, and on workstations.

Security patching challenges

Patching known vulnerabilities comes at a cost. Performing the activity at scale across your organisation could result in business disruption and regression defects.

When planning for this disruption in advance, you have two choices:

1. Spend additional time and resolve all known vulnerabilities over a longer patch maintenance window, at some point further in the future.

2. Move swiftly to resolve high and critical priority patches, in the near term.

Organisations that move swiftly have the best security posture. Security patching consumes time and resources both during and after the patching event, therefore it is prudent to prioritise the most important patches first. Focusing on patches that are critical or high priority mitigates the likely vector of breaches and data loss for your organisation.

Unintended side effects of patching

Larger organisations can invest in infrastructure and staffing to implement very streamlined patch management processes. Smaller organisations are usually more susceptible to disruptions associated with security patching processes, by virtue of having:

- An amorphous risk governance framework without clearly defined policies for validation, scheduling, and automation.

- A larger footprint of non-redundant systems.

- Limited, or inefficiently implemented security patching automations for critical and high priority vulnerabilities.

Without a mature risk governance framework, organisations of all sizes are vulnerable to business disruption through the security patching process.

Business disruption vectors

Business disruption associated with security patching comes in the form of:

- Human errors and staffing challenges associated with support, communication, and collaboration.

- Technology challenges that encompass hardware shortcomings, hardware failures, and their corresponding software errors.

Human error

If support staff within your organisation are unable to answer inquiries from your users related to a patch rollout, or in the absence of coordination between your organisation's system administrators and users, patching could be disrupted by users restarting their machines. If these restarts interrupt a security patch, the machine will likely boot into an unpatched state. The end result of incomplete patching due to human intervention is always the same: there will be a delay before the patching activity can successfully finalise across the organisation.

Staffing

Negotiating downtime is expensive and time consuming. Multiple groups within an organisation must agree and work in concert around server and workstation reboot cycles. The Ponemon Institute conducted a study which revealed only 36% of employees believe enough staff are available to apply patches fast enough.

Software

Software packages or their dependencies installed from unofficial channels, or compiled from source, complicate the patching process exponentially. The effects are further amplified when the person responsible for the installation is no longer available, or the implementation is poorly documented. In the worst case scenario, software from unofficial channels is an unknown variable, and is only discovered after Application Binary Interface (ABI) breaking changes bring the production workload to a halt.

Installing software from unofficial sources exposes you to the risk of forced upgrades. A forced upgrade is deliberately planned obsolescence of software, which has a ripple effect on its required dependencies. This translates to increased risks and costs for upgrading the machine, or migrating responsibilities from that machine, elsewhere.

Resource overload

Technology can begin to unravel when security patches are applied at scale. Security patching activities are limited by network and connectivity constraints. Compute constraints include available disk space on target machines, or coordinated rollouts for high-availability clusters. These network and compute constraints may extend the window required to successfully deploy the security patches.

In large organisations, patch management can cause network congestion. For example, many hosts might start downloading the same large patch, or bundle of patches, at the same time. This could consume excessive network bandwidth, or overwhelm the resources of the server responsible for serving the patches. Organisations should ensure their patch management infrastructure can avoid resource overload situations, by sizing the solution to meet the expected request volume, and staggering the patches over time.

Ageing hardware is vulnerable to failure during reboots. Full reboots push systems to 100% utilisation during the power-on self-test (POST) sequence, which is particularly stressful for older hardware. The adverse impact of failures is amplified if ageing hardware is non-redundant. System administrators will embark upon a lengthy replacement process for non-redundant hardware, as opposed to a shorter recovery process that would have been available if redundant machines were available. There is always the risk of patching activities extending beyond the estimated time, particularly for non-redundant systems.

Technology hiccups can impact patching activities in a variety of ways, and result in additional unplanned operational expenses for an organisation. Patch management solutions should conform to your risk governance framework to avoid and minimise disruption.

Regression Defects

Bugs sometimes find their way into software at inopportune times. Bugs introduced within a security patch traumatise system administrators, and encourage poor security hygiene. When a security patch feels just as harmful to the production workload as the vulnerability, by causing downtime, organisations begin feeling internal pressure to lengthen security patching windows.

Unidentified regression defects from security patching expose the lack of testing with phased patch rollouts, and poor validation of security patches. If an organisation's response to an unsuccessful security patch is to lengthen the patching interval, the problem has been delayed rather than solved.

Let us analyse a package where a security patch introduced a bug. If we take the python3-apt package as an example, the `apt policy` output sheds light on an available security patch and follow-on bug fix on Ubuntu 20.04:

```
1. $ apt policy python3-apt
2. python3-apt:
3.   Installed: 1.9.8
4.   Candidate: 2.0.0ubuntu0.20.04.8
5.   Version table:
6.      2.0.0ubuntu0.20.04.8 500
7.         500 http://archive.ubuntu.com/ubuntu focal-updates/main
           amd64 Packages
8.      2.0.0ubuntu0.20.04.3 500
9.         500 http://archive.ubuntu.com/ubuntu focal-security/main
           amd64 Packages
10.     2.0.0 500
11.        500 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages
12. *** 1.9.8 100
13.        100 /var/lib/dpkg/status
```

The `***` on Line 12 denotes the current installed version, which is also conveyed on Line 3. Line 4 identifies the latest version of the package, which is recommended for the system. Within the version table, we see version 2.0.0 available in the main pocket for Ubuntu 20.04. Subsequently, an update was made available in the security pocket. A minor version bump on line 6 indicates no new features were introduced, but seeing the package in the updates pocket confirms the fact that a bug fix patch was published for the security update.

It is possible to add additional colour to this information by evaluating data feeds for these software patches.

Introducing new vulnerabilities

Besides unwelcome bugs, security vulnerabilities that didn't exist before may suddenly exist after the patching cycle is complete. Software patches should always be followed by fresh vulnerability scans.

Patching activities may cause unintended long term side effects in the absence of a risk governance framework, with controls requiring vulnerability scanning after any package modifications. A common example is the installation of a security patch which inadvertently alters existing security configuration settings, or adding new settings with inappropriate defaults. In the process of fixing the

original vulnerability, a new security problem may manifest. Organisations should be capable of detecting these sorts of side effects, such as changes to security configuration settings caused by patch installations.

Patch management solutions should conform to your risk governance framework to avoid and minimise disruption. Now that we have discussed the most common challenges organisations face when deploying security patches, let's learn about tracking security vulnerabilities. Knowing where and how to find actionable data about security vulnerabilities empowers Chief Information Security Officers (CISOs) to implement a bulletproof risk-based patching strategy that aligns with their organisation's needs.

Track security vulnerabilities

Tracking security vulnerabilities effectively is the first step towards ensuring security and uptime.

Security vulnerability reports

It is important to know who the key players are when vulnerabilities are identified, communicated, prioritised, and patched. **Common Vulnerabilities and Exposures (CVEs)** have been recorded by Mitre since 1999, and are reflected in Canonical's searchable CVE reports dashboard. A CVE contains information about the impacted product's name, its version, and the name of the vendor. If security vulnerabilities stem from shared libraries, a separate CVE is assigned for each vendor affected. Canonical's CVE reports show recent CVEs for software that can run on Ubuntu, along with a priority rating from Canonical.

Common Vulnerability Scoring System Calculator

The Common Vulnerability Scoring System (CVSS) is an open standard used to compute vulnerability severities. This score is calculated through a function consuming Base Score Metrics, Temporal Score Metrics, and Environmental Score Metrics. The Base Score Metrics are derived from Exploitability Metrics and Impact Metrics. The CVSS score is a numerical value between 0 and 10.

- A Base Score of 0 has a negligible severity
- 0.1 to 3.9 has a low severity
- 4.0 to 6.9 has a medium severity
- 7.0 to 8.9 has a high severity
- 9.0 to 10.0 is critically severe

The Exploitability Metrics are calculated by choosing one answer for each of its components, reflecting the attack vector, attack complexity, privileges required, user interaction, and scope. Similarly, Impact Metrics and Impact Subscore Metrics are calculated by choosing one answer from each of its components, reflecting confidentiality, integrity, and availability impacts and requirements.

Environment Score Metrics

Exploitability Metrics*	Attack Vector	Attack Complexity	Privileges Requiered	User Interaction	Scope
	• Network • Adjacent Network • Local • Physical	• Low • High	• None • Low • High	• None • Required	• Changed • Unchanged

Impact Metrics*	Confidentiality Impact	Integrity Impact	Availability
	• None • Required	• None • Low • High	• None • Low • High

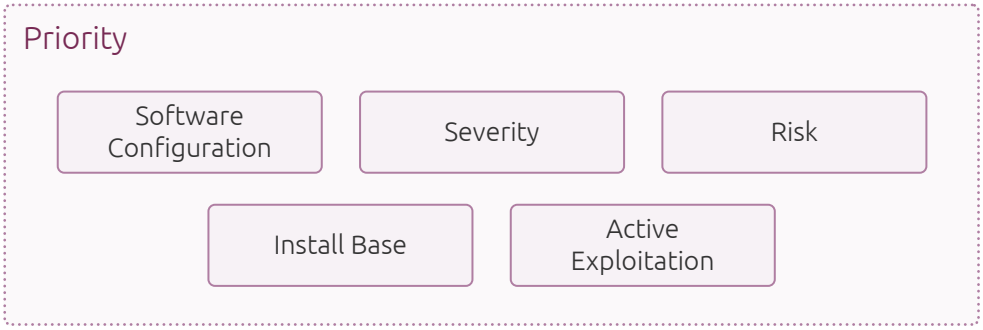
Impact Subscore Metrics	Confidentiality Requirement	Integrity Requirement	Availability Requirement
	• Non Defined • Low • Medium • High	• Non Defined • Low • Medium • High	• Non Defined • Low • Medium • High

** Base Score Metrics*

The National Vulnerability Database maintained by the US Department of Commerce provides a useful calculator that demystifies the scoring system.

Canonical's vulnerability priority score

Canonical assigns vulnerability priority based on many factors, including but not limited to severity, risk, install base, software configuration, and active exploitation.



When a security patch for a CVE aligns with Canonical's ongoing efforts to proactively improve security features in Ubuntu, this is reflected in Canonical's priority rating. The priority Canonical assigns to a CVE aggregates multiple data points to produce the final rating, and the CVE's priority will use the familiar scale of Negligible, Low, Medium, High, and Critical to convey urgency.

Canonical's analysis consumes severities, alongside other data points, and publishes a priority rating which is conveyed on a scale of:

- Negligible
- Low
- Medium
- High
- Critical

When a security issue is fixed in an official Ubuntu package, an official **Ubuntu Security Notice (USN)** is posted on Canonical's USN dashboard. CVE numbers are used to communicate about specific security vulnerabilities, and their severities are scoped to a specific software package. USNs will often address multiple related CVEs within a single grouping.

Returning to our earlier python3-apt example, on December 9, 2020 a software update was available for download from the security pocket. Security patch USN-4668-1 resolved the medium priority security vulnerability CVE-2020-27351. Bug fix patch USN-4668-2 was published on December 10, 2020, and was available for download from the updates pocket.

Knowing how vulnerabilities are identified and tracked is a prerequisite for establishing a sensible risk based vulnerability management and patch cadence. Canonical's priority classification in each CVE provides CISOs with a shortcut for evaluating and mitigating security vulnerabilities.

Value proposition of USNs

USNs have an awareness of follow-on regression bug fix patches which may be associated with earlier security patches. Security patching tools which are USN-aware will benefit from this awareness.

CISOs are responsible for determining patch priority, based on the exposure their machines have to each security vulnerability. Risk is influenced by how accessible the vulnerable machines are. A security vulnerability for web server software on a fleet of web servers poses a higher risk than the same vulnerability on an air gapped machine with limited physical and network access. Install base can amplify priority. If a vulnerability allows a single device to emit a small number of network packets in a particular direction, it may seem benign. If the install base is massive enough, this vulnerability could pose a significant distributed denial of service threat. Active exploitation is an important metric to evaluate when determining priority, because attacks require time, effort, and expertise for cybercriminals. Once the blueprint for a successful attack has been established, the time required to successfully attack new targets is dramatically reduced. Actively exploited vulnerabilities are very dangerous for exposed organisations.

Vulnerabilities which can be exploited in tandem produce an additional layer of complexity, therefore CISOs must have an understanding of exploit chains. Exploit chains are cyberattacks that leverage vulnerabilities from multiple exploits to compromise a target. Knowing the software configurations of related systems across your network is essential when preventing or contending with exploit chain attacks.

Every vulnerability scanner and patch management system is going to aggregate and present data by CVE number. Canonical's systems management solution, Landscape, is a scalable solution that manages tens of thousands of instances from within a single instance. Machines can be divided into cross sections by way of hierarchical access groups, user defined tags, and custom search parameters of hardware specific or software specific metadata. Landscape identifies porous attack surfaces that are missing security patches, and presents an actionable view of your estate, with security vulnerabilities by CVE and USN. Landscape can be leveraged to deploy security patches at scale, and by leveraging the capabilities of Ubuntu Pro Client, Landscape can perform vulnerability scanning that is inclusive of unpatched CVEs.

Beyond providing information about security vulnerabilities before the patch cycle, USNs provide value after the patch cycle has completed. If regression defects are identified in security patches, USNs keep track of any follow-on bug fix patches that need to be applied.

Risk-based patching

Performing accurate scans, rolling out patches in phases, balancing usability and security, and enabling automation are some best practices for managing risk, over time.

Organisational policies may inform patch cadence, but what is required in reality is much more nuanced. Organisations with scheduled maintenance windows should follow a risk-based vulnerability patch cadence. It is possible to reveal which vulnerabilities are actually exploitable, by relying on Canonical's priority ratings for at-a-glance insight into threat intelligence and attacker activity. CISOs and system administrators familiar with their own internal asset criticality can use this information to minimise operational risk.

Risk assessments inform automation and provide visibility into where risk gaps exist. These assessments should be conducted regularly, and remediations should be prioritised in accordance with a governance framework. A risk assessment should include a complete inventory of hardware and software assets, and identify shadow IT. Shadow IT is the use of IT hardware and software without knowledge of the security group responsible for the network. A successful risk assessment will conclude with a shared understanding of the cybersecurity risk to organisational operations, assets, and individuals. This shared understanding should lead to changes in controls to remediate identified risks.

- **Predictive controls**: proactive steps to remain aware of cybercriminal activities outside of the organisation.
- **Preventive controls**: proactive steps to stop or prevent harm, by way of passive and active monitoring of networks, file systems, and privilege management.
- **Detective controls**: reactive steps to identify threats that have already occurred.
- **Corrective controls**: restorative steps to return a system or process back to the state prior to the detrimental occurrence.

When implementing a risk based approach to security patching, it is necessary to define a fixed cadence for monitoring and acting upon these four controls. Your organisations' risk governance framework should clearly state that CVEs with a stronger chance of exploitation should be patched promptly, outside of the scheduled patching windows. The majority of vulnerability exploits are ones known by security and IT professionals at the time of the incident.

It is possible to focus on vulnerabilities that are actually exploitable by using Canonical's CVE priorities to serve as guidance for threat intelligence and attacker activity. Align Canonical's CVE priorities against your internal asset criticality to make risk-based security patching decisions.

Accurate vulnerability scans

Armed with detailed information about security vulnerabilities and our priorities for patching them, CISOs need a consistently reliable means of identifying which vulnerabilities exist across their organisation. Rather than using the `apt` package manager to identify vulnerabilities, it is wise to invest in a vulnerability management solution that has a deep awareness of security vulnerabilities, beyond what versions of software are available for download in a security repository pocket.

Repository configurations are as brittle as the number of administrators that have access to the machine. It is possible, particularly in environments where organisations mirror repositories internally, that the view a machine has of its available security patches is outdated or incomplete, and not representative of its true security patching posture. The configured repositories could be stale, inaccessible, misconfigured, or your Ubuntu instance may not have ESM entitlements. In these circumstances `apt` may report that your machine is fully up to date, and no additional security patches are available.

The true security posture can only be ascertained through an analysis utilising the latest data published on security.ubuntu.com. A comprehensive vulnerability scan would surface security vulnerabilities which do not have available software patches, and this is beyond the scope of `apt`'s capabilities.

Landscape maintains an independent vulnerability database of each Ubuntu distribution, and provides an external analysis of each machine's security status. Landscape can be used to provide deep insight into patchable and unpatchable security vulnerabilities on a per machine basis, comparable to the outputs when using the libopenscap8 package to scan USN OVAL data.

libopenscap8 is a package which contains the `oscap` binary, a utility that can parse structured machine readable data based on the **Open Vulnerability and Assessment Language (OVAL)**. Ubuntu's OVAL data feeds are published with OVAL vulnerability and patch definitions; this enables auditing for CVEs and identifies suitable patches by USN.

Generating security reports with oscap tool is incredibly easy:

```
$ sudo apt install -y libopenscap8
$ wget https://security-metadata.canonical.com/oval/com.ubuntu.$(lsb_
release -cs).usn.oval.xml.bz2
$ bunzip2 com.ubuntu.$(lsb_release -cs).usn.oval.xml.bz2
$ oscap oval eval --results $(lsb_release -cs).xml --report $(lsb_
release -cs).html com.ubuntu.$(lsb_release -cs).usn.oval.xml
```

OVAL is an industry standard, with a predictable and stable structure. This table summarises component specifications for the SCAP protocol at version 1.2.

- **Languages.** The SCAP languages provide standard vocabularies and conventions for expressing security policy, technical check mechanisms, and assessment results.
- **Reporting formats.** The SCAP reporting formats provide the necessary constructs to express collected information in standardised formats.

- **Enumerations.** Each SCAP enumeration defines a standard nomenclature (naming format) and an official dictionary or list of items expressed using that nomenclature.
- **Measurement and scoring systems.** In SCAP this refers to evaluating specific characteristics of a security weakness (for example, software vulnerabilities and security configuration issues) and, based on those characteristics, generating a score that reflects their relative severity.
- **Integrity protection.** An SCAP integrity protection specification helps to preserve the integrity of SCAP content and results.

SCAP Component	Description
Languages	
Extensible Configuration Checklist Description Format (XCCDF) 1.2	A language for authoring security checklists/ benchmarks and for reporting results of evaluating them
Open Vulnerability and Assessment Language (OVAL) 5.10	A language for representing system configuration information, assessing machine state, and reporting assessment results
Open Checklist Interactive Language (OCIL) 2.0	A language for representing assessment content that collects information from people or from existing data stores made by other data collection efforts
Reporting Formats	
Asset Reporting Format (ARF) 1.2 Asset Identification	A format for expressing the exchange of information about assets and the relationships between assets and reports
Asset Identification	A format for uniquely identifying assets based on known identifiers and/or known information about the assets
Enumerations	
Common Platform Enumeration (CPE) 2.3	A nomenclature and dictionary of hardware, operating systems, and applications, plus an applicability language for constructing complex logical groupings of CPE names
Common Configuration Enumeration (CCE) 5	A nomenclature and dictionary of software security configurations
Common Vulnerabilities and Exposures (CVE)	A nomenclature and dictionary of security-related software flaws
Measurement and Scoring Systems	
Common Vulnerability Scoring System (CVSS) 2.0	A system for measuring the relative severity of software flaw vulnerabilities
Common Configuration Scoring System (CCSS) 1.0	A system for measuring the relative severity of system security configuration issues
Integrity Protection	
Trust Model for Security Automation Data (TMSAD) 1.0	A specification for using digital signatures in a common trust model applied to other security automation specifications

Table 1: SCAP Version 1.2 Component Specifications
Source: Guide to Enterprise Patch Management Technologies, NIST Special Publication 800-40, Revision 3

For organisations looking for a minimal API to ascertain vulnerabilities on systems at scale, Canonical provides Pro Client. Pro Client is a command line tool, and the `pro` binary is bundled in the ubuntu-advantage-tools package. Pro Client provides detailed information about the state and status of an Ubuntu machine, and presents a wealth of information about exposed and patched vulnerabilities in machine readable (YAML and JSON) and human readable formats. Pro Client has an awareness of security patches available in premium repositories, and reliably surfaces security vulnerabilities when the `apt` package manager falls short. Landscape's vulnerability analysis can be enhanced by capturing outputs from Pro Client.

```
$ pro security-status --format yaml
```

When determining if OVAL or Pro Client is right for your organisation, it's important to note the differences:

	Pro Client	Oscap and OVAL data
Produces human readable output	In the terminal	As an HTML file
Produces machine readable output	JSON, YAML	XML
Identifies patched vulnerabilities	Yes	Yes
Identifies unpatched vulnerabilities	Yes	No
Works in air gapped environments	No, must reach security.ubuntu.com	Yes
Can apply security patches	Yes	No

Patch rollouts in phases

Security patches should be deployed through a phased approach. This approach allows process and user communication issues to be addressed at a smaller scale, before committing to deploying the patch broadly across the organisation. Once a small deployment is tested and validated, organisations should address the more difficult and complex environments.

Balance security against usability and availability

Earlier we outlined the challenge of contending with defects in patches, and surfaced the example of python3-apt. Solving this challenge entails testing patches before mass deployment.

Optimising for availability entails leveraging live patching solutions when possible. Leveraging self-updating snap packages and enabling Canonical's kernel Livepatch service for high and critical updates are avenues to minimise operating system reboots.

A seamless journey from Day 0 to Day 2

Canonical provides a turnkey security patching solution that works in even the most restrictive environments, through Landscape, Livepatch, and Snaps.

There are 3 stages of operations:

- Day 0 is the planning stage; determine the necessary resources and requirements to get up and running.
- Day 1 is the provisioning stage; deploy, install, and configure software.
- Day 2 is the maintenance stage; perform everything required to remain operational and safe.

Several of Canonical's products work in concert to bolster your risk governance framework. Landscape is particularly useful for Day 1 and Day 2 operations. Landscape provides outstanding system administration at scale for all versions of Ubuntu Desktop and Ubuntu Server, anywhere. Landscape has a scriptable API which integrates seamlessly with other Canonical products, third party software, and proprietary tooling within your organisation.

Reduce patch times and patch intervals

Landscape is the central point at which auditable scheduled and unscheduled security patching activities can be performed. Aligning your organisation to perform high priority and critical priority patches right away, and establishing a regular cadence for patching medium priority and below, will reduce patch times and also reduce the interval of time between patches. This approach will reduce the amount of time your organisation remains vulnerable to a patch, and ensures the most dangerous threats are mitigated as a priority.

Reduce the attack surface

Patching is hard, can break things, and takes time. Having a vulnerability management program can reduce your attack surface. By deploying file system monitoring and network monitoring solutions, you become a harder target for threat actors, as they try to gain leverage inside your environment. File integrity monitoring agents can be deployed en masse, and configured, using Landscape.

File integrity monitoring agents and network monitoring solutions are available as open source packages and proprietary closed source packages. Landscape can deploy and configure both types of solutions across your Ubuntu estate.

Identify the distributed sources for patches

Canonical ships two package managers with Ubuntu Desktop and Ubuntu Server: apt, and snap. Security vulnerabilities associated with packages installed with either of these package managers can be identified through OVAL data and Pro Client.

Snaps are app packages for desktop, cloud and IoT that are easy to install, secure, cross-platform and dependency-free. Snaps possess desirable security characteristics: they can run in isolation from the host system and have the means to independently self-update.

Packages installed from Canonical's repositories through apt have the benefit of undergoing strict centralised quality control. These packages can be audited, configured, installed, and removed from within Landscape. Security patches come from a designated security pocket, and bugfix patches come from a designated updates pocket. When Ubuntu LTS exits standard support and enters its **Expanded Security Maintenance (ESM)** window, all new security patches and their associated follow-on bug fix patches are served from that distribution's ESM repository.

Two types of patches can be applied live, through automation provided by Canonical, and do not require a system reboot to take effect:

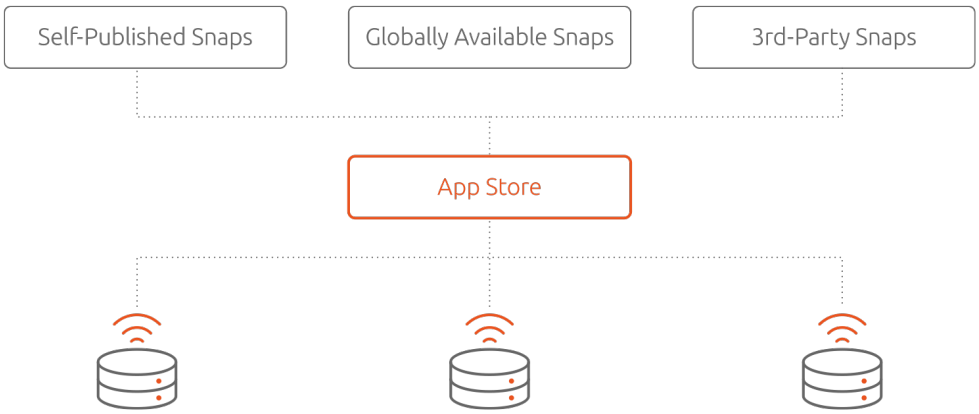
1. security patches through self-updating snaps

2. security patches delivered through Livepatch

Configure patch groups through these distributed sources

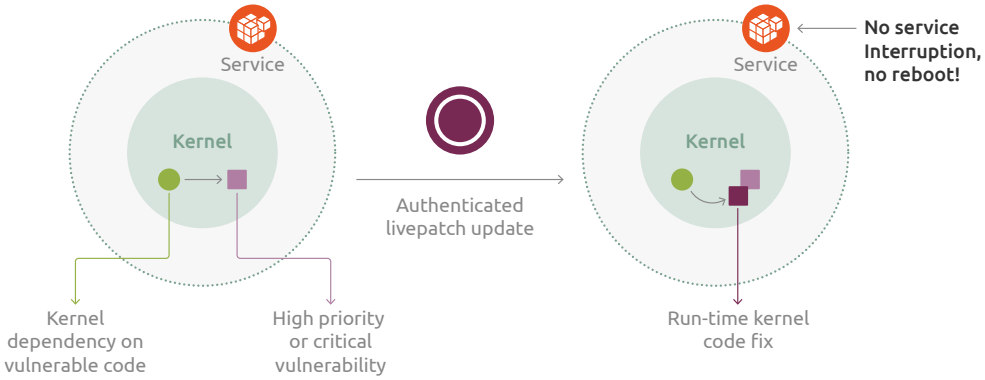
Snaps

If your organisation has policies around self updating software, it is possible to set up a Snap Store Proxy within your organisation. It will fetch packages from Canonical's snap store, but provide granular control over what revisions of snaps are installed on machines within your organisation.

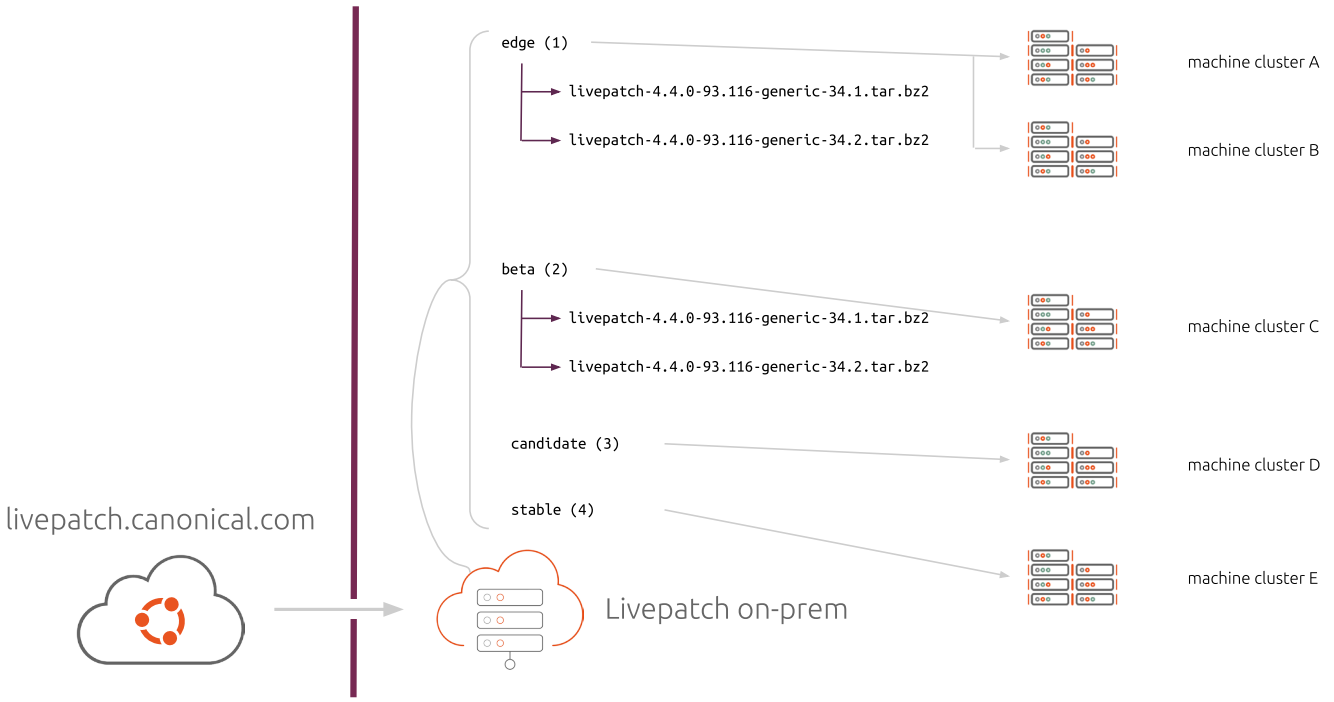


Livepatch

Canonical's Livepatch service provides a mechanism for stacking Linux kernel patches on running systems through an in-memory patching approach. Livepatch provides automatic high priority and critical priority kernel updates to machines, and applies these protections to live systems without the need to immediately reboot them. While these live patches are stackable, it's worth noting that the Livepatch service will not distribute kernel patches below a medium priority. Any outstanding medium or low priority security patches should be applied during your scheduled security patching interval.



Canonical's Livepatch on-premise solution also provides this capability. System administrators can divide their estate into tiers, and distribute kernel live patches in phased rollouts to specific groups of machines.



Apt

Landscape provides systems management capabilities at scale. Packages installed via `apt` can be added, removed, upgraded, rolled back, and monitored on demand. From within Landscape, it is possible to define automation policies and staggered update schedules to manage your estate over time. Upgrade policies can be applied to subsets of machines across an Ubuntu estate.

Any Linux kernel patches of medium priority and below do not get applied through Livepatch, and must be installed via apt. Rolling out these patches can be automated through tools like Landscape, Pro Client, and unattended-upgrades. To take control one step further, Landscape can install, configure, and run Pro Client, unattended-upgrades, snaps, and any other software to further your security patching objectives.

Summary of tools and services

Thanks to Canonical's robust tooling, system administrators have choices when it comes to deploying security patches across their Ubuntu estate. System administrators have a choice to patch their Ubuntu estate by software package name, security vulnerability's CVE number, or the security vulnerability's USN number. Canonical also provides tooling for on-demand patch analysis through systems which operate externally from the ones used to apply the patch. For example: a system could be patched by Landscape, and the patch can be verified through Pro Client. Alternatively, a system could be patched by Pro Client, and the patch can be verified through Landscape. OVAL analysis is also available, and this analysis can be performed on the machine itself, or it can be performed externally on a different machine.

When system administrators need to patch a specific software package by name, it is easy to search for that package in Landscape, and select the machines which should be upgraded. When multiple software packages tie back to a single patch, it is possible to resolve the CVE through Pro Client, and using Landscape to orchestrate the Pro Client activities at scale across your Ubuntu estate.

Your choice to use either Pro Client, Landscape, or both, is influenced by where you want the security analysis to happen, and if each machine you are managing has network access to security.canonical.com. While Landscape is more than capable of managing the security patching task on its own, the risk governance frameworks of your organisation may require additional or external analysis of each security patching activity. While Pro Client is limited to performing the security analysis on the machine it is installed on, Landscape can perform a security analysis within the Landscape server component for every machine it manages.

A machine is capable of performing an OVAL analysis as long as OVAL data for the distribution is available, and a software package manifest of all installed applications is available. Machines can self introspect their own security vulnerabilities by performing an OVAL analysis on themselves, or they can evaluate the security posture of other machines, without requiring any network connectivity. In network restricted environments, Landscape can take responsibility for patching, and OVAL analysis can be used as an additional, external validation mechanism to confirm the success of patching activities.

System administrators rarely have the luxury of security patching a homogenous fleet. Machines will have various versions of software, and may have varying degrees of security patching already applied. When multiple CVEs need to be resolved in a single maintenance window, and they all tie back to a single USN, both Landscape and Pro Client can facilitate pushing the USN patch, and auditing its application. In these challenging environments, auditing unscheduled patches uniformly across machines is convenient and easy with Landscape, Pro Client, and OVAL analysis.

Pro Client provides vulnerability scanning capabilities, and patching capabilities by CVE and USN numbers. This tool complements Landscape's ability to deliver updates on a package by package basis. Defining security patch schedules, deploying patches, and rolling back patches are Landscape's core competency. Landscape integrates with Pro Client seamlessly, and provides an auditable post-patch vulnerability analysis. Livepatch and snaps provide self updating capabilities for running systems, without requiring a reboot.

Next Steps

Beyond security and vulnerability patching, Landscape is an essential component of many organisations' broader compliance strategies. Self-hosted Landscape is free for limited personal or evaluation use. All machines with an active Ubuntu Pro subscription can use Landscape SaaS or self-hosted Landscape at no additional cost. Both editions of Landscape are included with Ubuntu Pro on AWS, Azure and GCP.

Canonical offers professional services for implementation, training, and consulting in connected, and air gapped environments. If you want to learn more, talk to us about Landscape and our professional services options:

Contact Us

Additional Resources

- https://docs.ubuntu.com/landscape/en/
- https://discourse.ubuntu.com/c/landscape/89
- https://ubuntu.com/landscape
- https://docs.ubuntu.com/snap-store-proxy/en/
- https://ubuntu.com/security/livepatch
- https://github.com/canonical/ubuntu-advantage-client
- https://ubuntu.com/security/oval

References

1. **https://www.gartner.com/smarterwithgartner/how-to-set-practical-time-frames-to-remedy-security-vulnerabilities**
2. **https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf**
3. **https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator**
4. **https://www.finra.org/sites/default/files/p602363%20Report%20on%20Cybersecurity%20Practices_0.pdf**
5. **https://www.servicenow.com/content/dam/servicenow-assets/public/en-us/doc-type/resource-center/analyst-report/ponemon-state-of-vulnerability-response.pdf**

Ubuntu