CANONICAL

Canonical NCSC Cloud Security Principles Assessment

June 2022

Purpose

This document provides responses to the NCSC-released TSR Vendor Assessment, which is a security framework the UK's National Cyber Security Center (NCSC) proposes to issue to network operators. The purpose of the NCSC's Vendor Assessment is to set best practices for ensuring the security of communications networks.

Who it's for?

This document was created for readers who are looking to assess Canonical-maintained products like Ubuntu, MAAS, Juju, Charmed OpenStack, Charmed Kubernetes, Charmed OSM, MicroK8s, LXD and others against NCSC requirements. It also aims to help mobile operators during the NCSC security audit, by providing responses for software provided by Canonical.

Scope

The document provides information relating to the NCSC criteria, responses from Canonical and additional links to dig deeper into relevant topics.

In case of additional questions or clarifications needed please contact us following instructions on www.ubuntu.com/contact-us

NCSC Criteria	Description	Canonical Response	Documentation
V.A: Product Lifecycle Management			
V.A.1: Product lifecycle process	Each product has a clearly defined lifecycle by the vendor. Vendors should have an End of Life Policy which covers details on how long the products will be in general support, extended support, and supported after End of life sales dates.	Canonical publishes new releases of Ubuntu on a regular cadence, enabling the community, businesses and developers to plan their roadmaps with the certainty of access to newer open source upstream capabilities. LTS or 'Long Term Support' releases are published every two years in April. LTS releases are the 'enterprise grade' releases of Ubuntu and are used the most. An estimated 95% of all Ubuntu installations are LTS releases. Every six months between LTS versions, Canonical publishes an interim release of Ubuntu. These are production-grade releases and supported for 9 months, with sufficient time provided for users to update, but do not receive the long-term commitment of LTS releases.	https://ubuntu.com/about/release-cycle
V.A.2: Software maintenance	Each product is maintained through its published life cycle. This maintenance, as a minimum, covers security fixes for the product.	Canonical supports packages that are provided as part of the Ubuntu main archive; these packages have undergone detailed review; potentially including a full security review depending on the software package and its potential security attack surface. Packages in Ubuntu main are supported inline with the published release cycle. Bugs are tracked using the Launchpad bug tracker. Stable updates are reviewed by a dedicated team to assess risk and ensure that the updates are compliant with the Stable Release Update policy. Security vulnerabilities are identified and resolved by the Ubuntu Security Team.	
V.A.3: Software version control	Each product has a version-controlled code repository which logs every code modification. This audit log will detail: what code has been modified, added, or removed; who made the change; when the change was made; and which version of the code has been built into the released product.	The majority of Open Source projects distributed in Ubuntu use version control to maintain a full version history of changes made to their codebase. A full history of package updates for every package in Ubuntu is maintained in Launchpad including changelogs and uploaders.	
V.A.4: Software releases	Each product goes through a rigorous software release cycle including internal testing before a version is released for general availability. Software will not be released if it does not comply with the Secure Engineering requirements detailed below. Each product should have regular external testing carried out on it by an independent third party.	Every Ubuntu development cycle is 6 months long; new versions of packages are uploaded during this development cycle prior to feature freeze, at which point only fixes or new versions deemed necessary should be uploaded. Release schedules are visible to the Ubuntu development community.	https://wiki.ubuntu.com/ReleaseSchedule

NCSC Criteria	Description	Canonical Response	Documentation
V.A.5: Development processes and feature development	There is one primary release train of the product. Forking of new versions is minimised. Where necessary, operator-specific functionality is provided as optional modules. Any new features are brought into the main product line during the standard development roadmap.	New features are planned and delivered aligned to the 6 month release schedule.	https://wiki.ubuntu.com/ReleaseSchedule
V.A.6: International release and forking	The vendor maintains a single, global version line for each product. There are a minimal number of other versions (ideally none). Specific localisation requirements such as language or cryptographic restrictions are not covered by this requirement.	Not applicable - there is only one Ubuntu release for the entire world.	
V.A.7: Use of tools, software and libraries	Third party tools (e.g. code compilers), software components and software libraries that are used within the product are inventoried. Third party tools critical to the security of the vendor's software are maintained throughout its lifetime.	Ubuntu as a distribution encapsulates a well defined set of software components and versions.	
V.A.8: Software Documentation	The vendor provides up-to-date and technically accurate documentation alongside new releases of the product. This documentation, as a minimum, shall detail how to securely configure, manage, and update the product.	Canonical maintains product documentation on the ubuntu.com website for all top-level products such as OpenStack, Kubernetes, Juju, MAAS and Ubuntu. Documentation is also contributed to the respective upstream projects for products such as Openstack, Kubernetes, Kubeflow, Ceph and OSM.	
V.B: Product Security Management			
V.B.1: Security culture	The vendor has a security culture which ensures that security principles are followed.	The Canonical Security Engineering team works across product engineering teams to perform threat analysis on key products including OpenStack, MAAS, Kubernetes.	
V.B.2: Secure Development Lifecycle	The vendor has a Software Secure Development Lifecycle to ensure that all software remains in a maintainable state. All development teams follow the same defined Secure Development Lifecycle processes. Having a Secure Development Lifecycle within a company also helps generate a culture of security awareness within development teams.	Ubuntu follows the SDLC process as part of the development process established by the community. Details of that process can be found on the Ubuntu Developer wiki.	https://wiki.ubuntu.com/UbuntuDevelopment
V.B.3: Internal component management	Any shared internal components or libraries are kept up to date and only the latest stable, supported version is used. These components and libraries are not be modified for specific builds and are supported for the lifetime of the product.	All components that make up Ubuntu main are managed as part of the distribution process.	
V.B.4: External component management	Only supported external components are used within a product. The vendor monitors the external component's changelog so that only the latest supported, stable version is used within the product. Additionally, the vendor monitors the external component's security advisories and pulls in any security fixes and integrates them into their product with a security update.	All components that make up Ubuntu main are managed as part of the distribution process.	

NCSC Criteria	Description	Canonical Response	Documentation
V.B.5: Unsafe Functions	There are no unsafe functions used within the vendor's released code.	Use of unsafe functions forms part of the main inclusion request process as part of any security review undertaken. Code security analysis tools are used for this review.	
V.B.6: Redundant and duplicate code	The vendor's source tree is maintained to a level that there is limited redundant or duplicate code.	Vendoring of code and duplication/forking of projects is discouraged in Ubuntu main; the main inclusion process specifically reviews and flags any of this type of activity.	
V.B.7: File structure	The vendor's source tree is maintained to a level where code complexity is minimised, and functions perform single, clear, actions.	All of the software components included in Ubuntu main will use typical file structures within their source code and will have codified package build processes.	
V.B.8: Debug functionality	There is no engineering debug functionality present within the vendor's released products	Debug functionality exists but is disabled by default for telecom operators' installations.	
V.B.9: Comments	The source tree has suitable and understandable comments through it, explaining what the code is for and how it performs its actions.	Commenting is encouraged in areas of code complexity in Canonical managed projects. Levels of commenting on other projects included in Ubuntu may vary and might not always follow the same standards.	
V.C: Protected development and build environments			
V.C.1: Segregation of development environment	Development environment is segregated from the corporate network and protected from the internet.	Segregation of the 'development environment' in distributed Open Source projects is not possible.	
V.C.2: Segregation of build environment	Build environment is segregated from the corporate network and protected from the internet. Very few people can make changes.	Launchpad builders where all components are compiled into Debian packages or Snaps are hosted on a dedicated OpenStack cloud which is isolated from other parts of Canonical's network infrastructure and is administered by a dedicated team.	
V.C.3: Build automation	Build environments are simple, and the build process is automated.	Debian packages make use of a managed build secure root image; Debian source packages detail dependencies required to build the package, all of which must be sourced from the Ubuntu or Ubuntu Cloud Archive repositories. Build environments are easily reproduced using developer tooling to reproduce build failures. Snaps build against an Ubuntu Core image using the snapcraft build tool; dependencies are detailed in the definition of the snap (snapcraft.yaml). For both distribution methods, developers can easily reproduce the build process outside of the secured build environment used for build and publication of release packages.	https://snapcraft.io/docs/build-and-staging-dependencies

NCSC Criteria	Description	Canonical Response	Documentation
V.C.4: Role-based access	Only developers have access to the internal code base, and access is controlled and limited based on role. Developers should only use specified versions of open source libraries, pulled from a central location within the vendor's development network.	RBAC is enforced in multiple ways for software projects that form part of Canonical's products; Upstream projects typically use VCS repositories with core developers having commit privileges; some projects may also not allow direct commits, requiring pull requests or reviews which are peer reviewed before approval subsequently committed automatically into the VCS repository. Packaging source is typically controlled in similar ways. Ubuntu developers with permission to upload to the Ubuntu archive build a source package which is signed by the developer and uploaded to Launchpad. which verifies the signature and then uses the secure build environment to build the binary packages.	
V.C.5: Code review	All code is independently reviewed prior to acceptance. Feedback processes exist.	A large number of Open Source projects have peer code review as a core part of their development ethos - this includes Ubuntu, MAAS, OpenStack and others. For example, OpenStack uses Gerrit for code review on all proposed changes, typically with pre-commit functional testing to support the reviewer.	
V.C.6: Repeatable builds	All builds of released software must be repeatable at a future date	Builds of Debian packages against an Ubuntu release are by their nature repeatable as the packages in a released version of Ubuntu should only receive stable release updates for bug fixes or security issues. This is an inherent feature of a Linux distribution such as Ubuntu. Packages have changelogs which detail any changes that have been made since the original release. Build logs of all builds are maintained on Launchpad with details of dependencies (including versions) used to produce the binary package builds across multiple architectures.	
V.D: Exploit mitigations			
V.D.1: Heap Protections	The vendor makes use of modern heap protection mitigations to help prevent heap-based memory corruption attacks against the product.	Heap Protector The GNU C Library heap protector (both automatic via ptmalloc and manual) provides corrupted-list/unlink/double-free/overflow protections to the glibc heap memory manager (first introduced in glibc 2.3.4). This stops the ability to perform arbitrary code execution via heap memory overflows that try to corrupt the control structures of the malloc heap memory areas. This protection has evolved over time, adding more and more protections as additional corner-cases were researched. As it currently stands, glibc 2.10 and later appears to successfully resist even these hard-to-hit conditions.	
V.D.2: Stack Protections	The vendor only ships executable code that has been compiled using modern stack mitigations.	All code built for Ubuntu is compiled using '-fstack-protector-strong' as part of the standard build flags.	

NCSC Criteria	Description	Canonical Response	Documentation
V.D.3: Data Execution Prevention	The vendor supports hardware-enforced data execution prevention for example DEP.	DEP is used by default in Ubuntu. This is done via the NX bit if the CPU supports it, or emulated via memory segmentation if the CPU does not support it.	https://wiki.ubuntu.com/Security/Features#nx
V.D.4: Address Space Layout Randomisation	The vendor only ships executable code that has been compiled using modern ASLR techniques.	Ubuntu uses ASLR on several levels.	https://wiki.ubuntu.com/Security/Features#stack-aslr
V.D.5: Memory mapping protections	The vendor's product will have no memory pages mapped by default as both "Writable" and "Executable". This excludes areas of the code required to do Just-In-Time code compilation.	There is no memory mapping enabled by default in Ubuntu.	https://wiki.ubuntu.com/Security/Features
V.D.6: Least Privilege code	The vendor follows a "least privilege" methodology when developing and executing code within their products. The vendor ensures that their product only runs at or requests the minimum privilege level required for it to fulfil its advertised purpose. If higher privilege levels are ever required, then the product only elevates privilege for the specific task.	Canonical products follow a "least privilege" methodology when developing and executing code within their products.	
V.D.7: Secure execution environment	The vendor has a product road map detailing when and how they plan to implement secure execution environments to enable execution of sensitive workloads on untrusted hardware.	Ubuntu already delivers secure execution environments. Also, the telco edge setup with MAAS, LXD and MicroK8s running on Ubuntu Core provides confinement based on Snaps to work with untrusted workloads.	https://ubuntu.com/blog/ibm-z-delivers-a-trusted-execution-environment-with-ubuntu-20-04-lts-and-new-single-frame-models
V.E: Secure Updates and Software Signing			
V.E.1: Software and firmware signing	Vendor's software and firmware is digitally-signed.	Packages in Ubuntu and the Ubuntu Cloud Archive are not directly signed. Instead the archive maintains indexes of packages and checksums in signed files which can be used by Ubuntu installations to verify the integrity of the packages downloaded for install or update.	
V.E.2: Signature verification	Software signatures are verified before binaries are executed.	By default any signature verification will result in packages not being installed. The public component of the archive signing key is a core part of the base operating system.	
V.E.3: Secure update	Updates are delivered via a secure channel that is mutually authenticated.	The nature of the design of the archive signing process is secure and can be used safely over http or https connections.	
V.E.4: Downgrade protection	Built-in detection capabilities alert whenever software is downgraded during an install process.	Any attempt to downgrade a package to an older version results in a warning and a prompt to the operator.	
V.F: Hardware roots of trust and secure boot			
V.F.1: Hardware root-of-trust	The equipment contains a hardware root-of-trust for identity and storage.	Not applicable as Canonical does not provide any hardware products.	

NCSC Criteria	Description	Canonical Response	Documentation
V.F.2: Secure Boot	Each product will support a secure boot process, initiated by the hardware root-of-trust (V.F.1) to bring the equipment into a known-good state on restart.	On Ubuntu, all pre-built binaries intended to be loaded as part of the boot process, with the exception of the initrd image, are signed by Canonical's UEFI certificate, which itself is implicitly trusted by being embedded in the shim loader, itself signed by Microsoft. On architectures or systems where pre-loaded signing certificates from Microsoft are not available or loaded in firmware, users may replace the existing signatures on shim or grub and load them as they wish, verifying against their own certificates imported in the system's firmware.	https://wiki.ubuntu.com/UEFI/SecureBoot
V.F.3: Securing JTAG	Each compute element on product will have JTAG access disabled	Not applicable as Canonical does not provide any hardware products.	
V.G: Security Testing			
V.G.1: Automated testing	Once developed, extensive security tests are automatically run against all versions of products.	Automated testing, which covers security aspects, is done by Ubuntu QA team, a follows the same process and is run against all versions.	https://wiki.ubuntu.com/Testing/Automation
V.G.2: Testing rigour	Developers cannot modify the build environment to hide or disregard build issues, or issues detected by automated tests. Failing builds are automatically rejected. Therefore, code does not create any compiler errors or warnings during build.	Launchpad builders where all components are compiled into Debian packages or Snaps are hosted on a dedicated OpenStack cloud which is isolated from other parts of Canonical's network infrastructure and is managed by a dedicated team.	
V.G.3: Security Testing	Security functionality is tested to demonstrate correct operation.	Before a security update is made available, it is tested against the vulnerability it aims to fix and for regressions. The Security Team uses the QA Regression Testing suite when performing testing. QA Regression Testing has information on performing tests, checklists, scripts and various other information to help with testing.	
V.G.4: Negative testing	Extensive negative testing is performed against every product release, including a wide range of potential failure cases, inappropriate message sequencing and malformed messages.	Negative testing is a part of the standard automated testing process as per VG1.	
V.G.5: Fuzzing	Fuzzing is performed against the product. The approach is complex enough to ensure that a high proportion of code is tested.	Searching for security vulnerabilities is usually referred to as auditing. The Ubuntu Security Team often performs audits on software before it is to be officially supported. Once vulnerabilities are found, the Security Team uses responsible disclosure to let others know about the issue. The Auditing page has more information.	https://wiki.ubuntu.com/SecurityTeam/Auditing
V.G.6: External testing	External security research teams perform testing against a selection of major product releases. Some of this testing is un-scoped.	See VG5	
V.G.7: Dynamic application security testing (DAST)	The vendor has a DAST solution integrated into the vendor's test process	ZED Attack Proxy or ZAP, coming from OWASP community is used for DSAT	

NCSC Criteria	Description	Canonical Response	Documentation
V.H: Secure management and configuration			
V.H.1: Product lock-down	The product can be locked into a secure configuration (E.G., vendor lockdown guidance, equipment-specific SELinux). Documentation exists to help customers perform lock-down.	AppArmor profiles used where appropriate.	https://wiki.ubuntu.com/AppArmor
V.H.2: Protocol Standardisation	The product can be configured to only use standardised protocols.	No proprietary protocols are used as part of Ubuntu and related Canonical operated open source projects.	
V.H.3: Management plane security	By default, the product is configured to only use modern, secure protocols on the management plane.	Ubuntu follows best practices with regards to TLS protocols and cipher suites.	
V.H.4: Management access	By default, access to the management plane is user-based and certificate asymmetric-key-based.	MAAS and OpenStack access is user-based, and it can be LDAP integrated.	
V.H.5: No unencrypted protocols	Secure protocols are used whenever possible (e.g. SSH and HTTPS). If an unencrypted protocol is enabled, and a secure alternative exists, the product warns the administrator, and provides the option to create a security alert.	Secure protocols are used whenever possible in Canonical solutions.	
V.H.6: No un-documented administrative mechanisms	The product does not have any undocumented administrator accounts. Examples include, but are not limited to, hard coded passwords, access key pairs (SSH keys) or otherwise administrative access tokens.	Canonical products have no undocumented administrative mechanisms which can be easily confirmed based on the open source nature of all products.	
V.H.7: No un-documented administrative features	The product does not have any undocumented administration features	Canonical products have no undocumented administrative features which can be easily confirmed based on the open source nature of all products.	
V.H.8: No default credentials	No default passwords are left on the device after the initial set up. For clarity, this also means there are no administrative accounts coded into the vendor's software.	No default credentials are created as part of an Ubuntu installation apart from the 'ubuntu' user. By default this user has no password set and can only be accessed via SSH with predefined authorized SSH keys.	
V.H.9: Best Practice Guidance	The vendor is explicit about the threats to the equipment that they have sought to mitigate, and those they have not. The vendor provides detailed configuration and notes on how the equipment can be protected in networks.	Best practices guide exists and is available online.	https://docs.openstack.org/security-guide/ https://wiki.ubuntu.com/BasicSecurity
V.J: Vulnerability and Issue Management			
V.J.1: Issue tracking and remediation	The vendor has a process for issue remediation. This ensures the vulnerability is resolved in all impacted products. Vulnerabilities are patched within appropriate timeframes.	Canonical has a process for issue remediation which is public and transparent. Security notices are released for all packages.	https://ubuntu.com/security/notices

NCSC Criteria	Description	Canonical Response	Documentation
V.J.2: Issue comprehension	For issues, the vendor identifies the root cause analysis of the issue and is able to detail the origin of the vulnerability.	In contrast, the priority value assigned by the Ubuntu Security team is designed to capture these and other elements so that it can be used as an effective measure to prioritise security software updates taking into account every Ubuntu instance – including server, desktop, cloud, and IoT. Vulnerabilities that affect the largest number of Ubuntu installations and which present the largest risk (by say being remotely exploitable without any user input, etc.) are prioritised critical or high. Those which affect only a small number of users and might require user-input or might only cause smaller effects such as a denial-of-service may be prioritised as a medium, low or negligible. This CVE prioritisation is done on a case-by-case basis for each vulnerability, and since a given vulnerability might apply to more than one package in the Ubuntu archive, this can be assigned further on a vulnerability-per-package basis as well.	
V.J.3: Vulnerability reporting	The vendor provides a publicly advertised route for disclosure of security issues that links into their vulnerability management process.	The Vulnerability reporting process is public and described in the Ubuntu FAQ available online.	https://wiki.ubuntu.com/SecurityTeam/FAQ
V.J.4: Issue transparency	The vendor is transparent about their patching of security issues.	The Ubuntu Security team manages their own CVE database to track various CVEs against the software packages within the Ubuntu archive. As part of this process, each day the team triages the latest public vulnerabilities from various sources, including MITRE, NIST NVD and others.	https://ubuntu.com/security/cve

Ubuntu