Technical White Paper

UBUNTU CORE

Cybersecurity Analysis

Ubuntu Core is a complete open source solution for a predictable, reliable, and secure operating system specifically targeted at Internet of Things (IoT) devices and highly scalable container deployments. As a comprehensive, secure ecosystem, Ubuntu Core solves many of the challenges associated with traditional Linux distribution models, while providing developers with unprecedented flexibility and the ability to truly continuously integrate and deploy to any number of devices.

This white paper provides an independent, third-party evaluation of the security provided by Ubuntu Core. The statements in these pages are data driven, built upon extensive testing of the ecosystem, including the overall architecture and design, authentication controls, deployment to end-user devices, and execution of custom device applications via snaps.

Based on the analysis, Ubuntu Core represents a well-balanced approach to securely deploy and maintain applications on IoT devices, embedded systems, or similar large-scale deployments.

KEY TAKEAWAYS

- Ubuntu Core represents a secure, structured ecosystem for large scale IoT device deployments and software lifecycle management.
- There are many inherent benefits to adopting Ubuntu Core as the platform for large scale CI/CD device deployments.
- A thorough review of the entire ecosystem resulted in no critical security findings.

Version 1.1
January 14, 2020

Introduction

Ubuntu, as the world's most widely deployed Linux distribution, is embraced by users, developers, and manufacturers alike as the easy-to-use, feature-rich de facto Linux standard. Recognizing the advent of Internet of Things (IoT) devices and large container deployments, Canonical has created an open source, purpose-built distribution for this new world: Ubuntu Core.

Canonical has provided the community with substantial documentation and supporting materials explaining the Ubuntu Core architecture and approach, including details of the control points that provide security protections. To remain consistent with industry practice, however, security architecture and controls should always be reviewed by a qualified, independent party to identify strengths and residual risks.

This white paper provides an independent, third-party evaluation of the security architecture and controls provided by Ubuntu Core and its ecosystem.

ABOUT RULE4

Rule4 is a highly credentialed global provider of cybersecurity and emerging technology professional services. Combined, its experts have more than a century of experience in the field, and its team is recognized industry-wide for leveraging this experience to evaluate risk and solve complex problems in a practical way.

Rule4 has amassed credentials that include CISSP, CSSA, GIAC GCFA, CISA, ISSMP, HCISSP, ISSAP, GSNA, and OSCP certifications. Its engineers are led by two co-authors of the *Linux and Unix System Administration Handbook*, now in its fifth edition.

As a Certified B Corporation, Rule4 proudly demonstrates the organization's commitment across the board to first do what's right.

Viewpoint on Ubuntu Core

The advent of Linux has brought transformational change to the technology landscape, offering a rich set of abstractions and tools for the general-purpose computing market. Unfortunately, the historic focus on supporting such a wide array of user-focused features has produced Linux distributions that are easy-to-use and incredibly powerful, but not well-suited for purpose-specific uses such as appliances, IoT/Industrial IoT (IIoT) devices, and other situations where an embedded operating system is desired.

Specialized embedded operating systems have been available for many decades, but typically have suffered as "closed" products where functionality was limited and enhancements were highly dependent on the vendor. Even more problematic is the lack of fleet management functionality — typically, the embedded OS was installed when the device was shipped, and barring some herculean effort, that same version and functionality were likely still on the device when it went to its grave. This may have been acceptable in the days of non-networked devices, but we're well past that.

It's clear that Ubuntu Core was thoughtfully architected with this knowledge and real-world experience in mind. It strikes the balance of benefiting from the flexibility of the rich Linux ecosystem, but in an open source framework that provides security, scalability, and manageability. These essential attributes frame the benefits provided by Ubuntu Core.

Hardened by design

Ubuntu Core takes a security-first approach, ensuring that security is built in throughout the entire application and device lifecycle. While most of the Ubuntu Core attributes directly support security in some manner, the architecture itself establishes a secure foundation that is easily built upon. As illustrated in Figure 1, a minimal OS, the kernel, and device drivers are packaged and installed as snaps, as are gadget-specific applications. At runtime, individual applications are rigidly sandboxed via a policy-based system that restricts access to the filesystem, network interfaces, system calls, and other standard Linux facilities. This approach provides an extraordinary amount of fine-grained security control that can be used to ensure that both the device and any associated data is adequately protected. Out of the box, Ubuntu Core provides the ability to easily deploy security updates across the application, gadget, and base system.

Complexity management and reduction

Through the standardization of application and feature deployments via snaps, combined with centralized management through a brand-specific snap store, Ubuntu Core greatly reduces overall complexity in environments where fleets of devices need to be deployed, secured, managed, and updated. This approach enables global device management capabilities, whether it be within a large enterprise or across a large, distributed customer base.

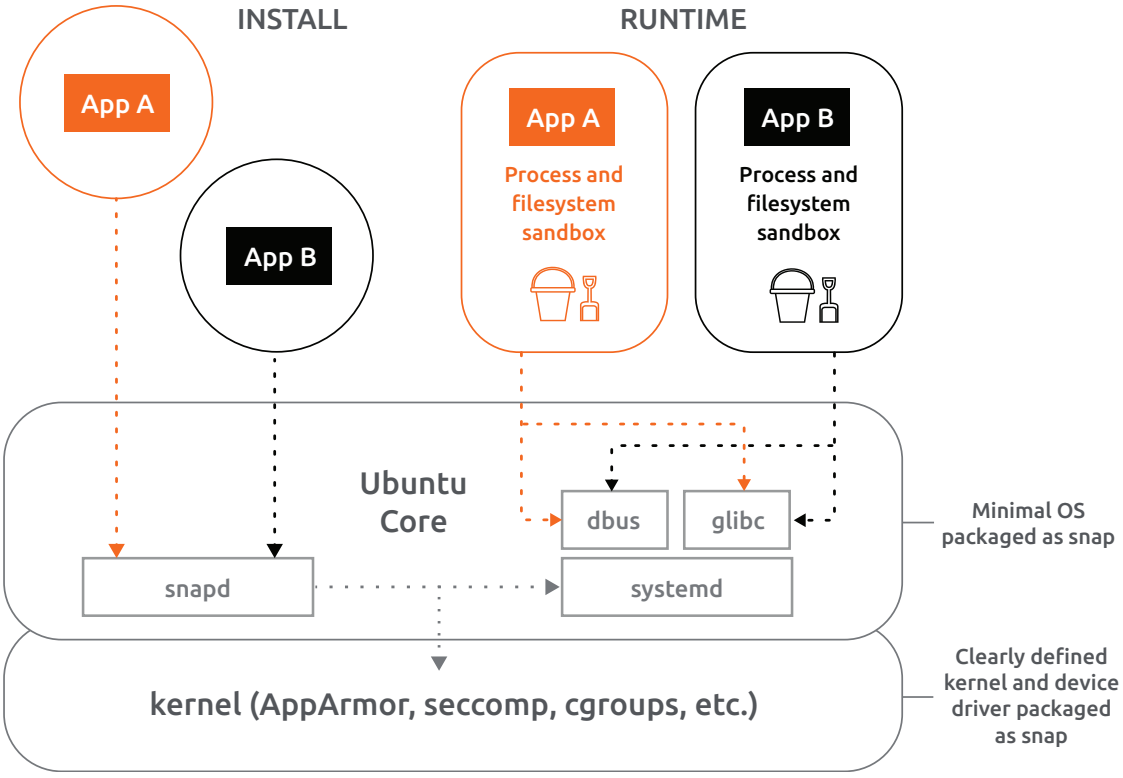

Figure 1: Ubuntu Core OS and package installation and execution

DevOps and SOAR alignment

The DevOps and Security Orchestration, Automation, and Response (SOAR) movements emphasize the need for modularity and automation, with integration spread across multiple technical disciplines. This approach creates an environment where consistency, repeatability, and security are driven through a standardized, automated toolset. The Ubuntu Core ecosystem slots into a continuous integration/ continuous deployment (CI/CD) pipeline perfectly. End-to-end application and lifecycle management are provided by Ubuntu Core as illustrated in Figure 2.

High velocity deployment

By providing tools that leverage existing Linux projects, Ubuntu Core enables faster prototyping which ultimately reduces time-to-market. Developers can now bring the knowledge, innovation, and power of the entire Linux community to bear on the problem at hand. This means support hardware and integrations in addition to security fixes and enhancements.

Rapid remediation

One of the most significant challenges facing the embedded systems and IoT device market is long-term lifecycle management. Maintaining the security profile of a fleet of devices in the field requires a mechanism to easily and rapidly deploy patches and upgrades as new vulnerabilities are discovered. Using the snap deployment infrastructure, Ubuntu Core developers can rapidly deploy targeted patches to affected devices without any end-user involvement.

Security offload

As the complexity and utility of technology has increased, so too has the scope of the environment that needs to be secured. By utilizing the rich existing code base for Linux, the burden of identifying and patching vulnerabilities in common elements such as the kernel, network services, and other common tools is effectively offloaded to the community, such that the Ubuntu Core developer need only focus their security mindshare on their purpose-specific application. This results in an overall system that has been vetted by thousands of developers and security practitioners worldwide, increasing the overall trustworthiness of the device.

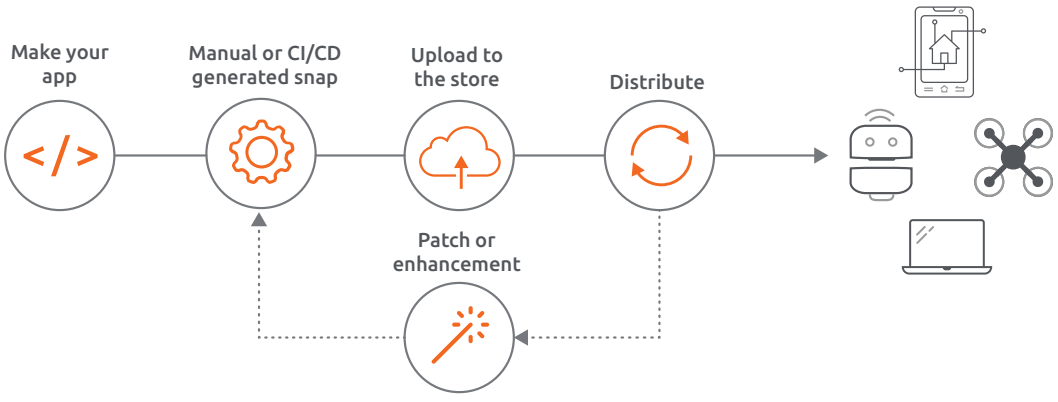

Figure 2: Application lifecycle in Ubuntu Core

UBUNTU CORE SANDBOXING

The primary tenet of Ubuntu Core security is application sandboxing. Canonical's *Ubuntu Core Security Whitepaper* provides an overview of the components that facilitate the sandboxed environments within Ubuntu Core; these are summarized in the table below for reference.

You can learn more about sandboxing and other Ubuntu Core technical controls, including how to configure them (where applicable), within these resources:

- *Security and Sandboxing* – https://docs.ubuntu.com/core/en/guides/intro/security
- *Ubuntu Core Security Whitepaper* – https://assets.ubuntu.com/v1/66fcd858-ubuntu-core-security-whitepaper.pdf
- *Snap Security Confinement* – https://snapcraft.io/blog/where-eagles-snap-snap-security-overview

AppArmor – AppArmor is Ubuntu's Mandatory Access Control (MAC) system, which ensures kernel-level enforcement of programs and processes to a limited set of resources. Application confinement in Ubuntu Core is such that the child process will inherit the parent's label and therefore policy. AppArmor restricts processes running either as root or non-root, and confinement policy is provided via profiles loaded into the kernel.	Namespaces – A facility provided by the Linux kernel that allows separation of processes such that they cannot see or access resources from another namespace. Several namespaces exist, such as file, network, and mount. Ubuntu Core uses a mount namespace to implement a per-snap /tmp directory in addition to sharing content between snaps and the system.	Seccomp – User space programs that need to interact with the hardware do so via kernel syscalls. The launcher will set a seccomp filter for the program before executing it to limit the syscalls the process may use. Child processes inherit the parent's seccomp filter, and they can make the filter more, but not less, strict.
Traditional Permissions – The Linux kernel enforces Discretionary Access Controls (DAC) via traditional UNIX "owner" permissions and Linux kernel capabilities. For app snaps on Ubuntu Core, services run as root and therefore traditional permissions alone don't play as important a role in the confinement of services.	Control Groups – Cgroups group processes for resource limiting, prioritizing, accounting, and more. Ubuntu Core currently uses the "devices" cgroup for hardware device access controls for hardware assignment.	devpts newinstance – The Linux kernel provides pseudoterminal (PTYs) functionality for login sessions and TTY capabilities. Ubuntu Core configures the devpts filesystem in multi-instance mode and mounts a new devpts instance per command to prevent snooping and input injection via /dev/pts.
Ubuntu Hardening – Canonical-supplied kernels have the kernel hardening benefits of classic Ubuntu kernels including ptrace scoping, symlink restrictions, and hardlink restrictions. Applications using the Ubuntu Core base system libraries and interpreters, as well as applications built with the Ubuntu toolchain or bundling debs from the Ubuntu archive (e.g., using snapcraft), benefit from the same toolchain and glibc hardening protections available to classic Ubuntu and the Ubuntu Core base system.		

Testing Approach and Findings

Independent third-party cybersecurity review, testing, and validation are important components of a comprehensive cybersecurity program. The testing activity described here was performed independently and provides an unbiased third-party perspective on risks within the Ubuntu Core ecosystem.

A disciplined, methodical, and comprehensive analysis of Ubuntu Core was used to validate the strengths of its cybersecurity controls and identify any potential deficiencies in its architecture. The cybersecurity profile of the entire Ubuntu Core ecosystem was built using a combination of detailed threat mapping and hands-on technical testing of controls and behaviors.

At a high level, the testing approach followed this process:

1. Scope bounding, verification, and final definition based on objectives
2. Evaluation approach and methodology development
3. Control point collection and development
4. Control point testing and findings documentation

Scope of review

The potential use cases for Ubuntu Core are limited only by the creativity and imagination of those using it as a baseline for projects or products. Developers must retain responsibility for implementation decisions that may alter the expected behavior of a system, but that's predicated on knowledge and understanding of the baseline capabilities on which that customization occurs. Figure 3 depicts the boundaries used as the basis for this review. Elements colored orange were defined as in scope. As implied by the diagram, Canonical infra-structure, policy, and procedures were not the focus of this review.

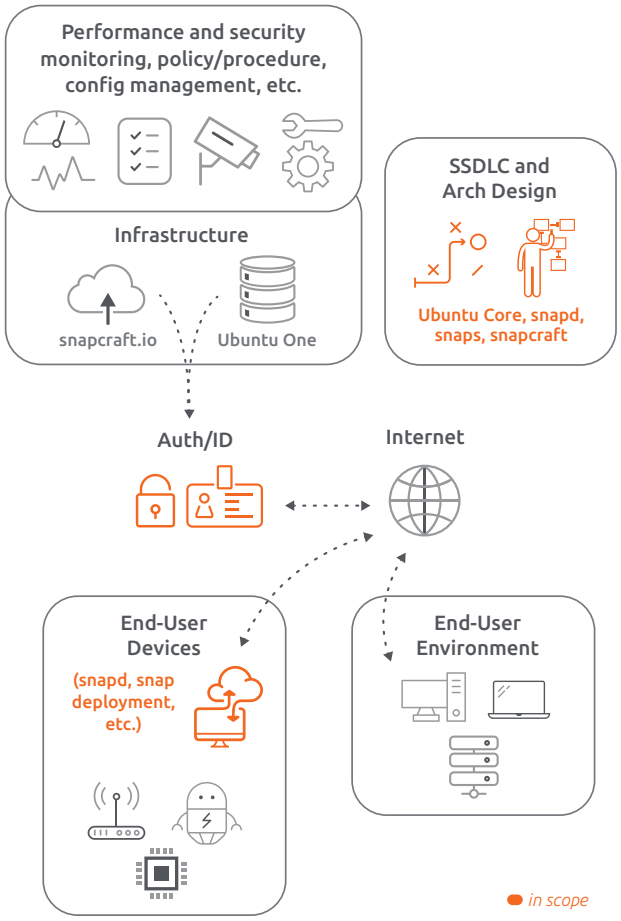

Figure 3. Testing scope

Evaluation methodology

The approach for this Ubuntu Core and ecosystem review shares some similarities with Common Criteria (CC) evaluation such as that performed on Ubuntu 16.04. Formal security specifications haven't been publicly developed or released by Canonical for Ubuntu Core in support of CC certification; however, Canonical provides documentation that helps illuminate the security controls and design of the Ubuntu Core system.

These sources of documented cybersecurity control points, both explicit and implied, were

used as a baseline for technical testing. In some cases, control points were derived from critical consideration of the intent, control, or overall objective of the area of review. The methodology is depicted in Figure 4.

Control point data sources

Multiple data sources were used to build a control point inventory. The primary source was the *Ubuntu Core Security Whitepaper*, though several other references (including official blog posts) were used to gather control point baselines. A complete list of sources is presented in the appendix.

Source material was then carved into domains of review:

- Ubuntu Core OS (kernel + utilities)
- Snap package/container model
- Snapd and snap deployment
- Brand snap store
- Authentication, identification, and authorization
- Public snap control/protections
- Trust model/ecosystem
- Threat modeling and risk allocation

Control point testing

Threat vectors for testing were developed through a threat modeling exercise that identified outcomes if the control point were found to be weak. Testing was then performed to determine the efficacy of the control point and to identify any potential exploitable vulnerabilities. In total, 136 threat map entries were developed, reviewed, and tested.

To better understand the threat chain, consider an example: A statement from the *Ubuntu Core Security Whitepaper*, "The snap's security policy does not allow modification of the security sandbox in which it runs," is treated as an explicit control point. A threat vector based on this control point would be "Security policy bypass enables modification of the security sandbox." From there, the appropriate test is to attempt to identify mechanisms or approaches by which policy could be bypassed and the sandbox modified.

In every case, testing focused only on claimed cybersecurity control points and was not intended to pursue each and every potential failed control point or threat vector to exploitation.

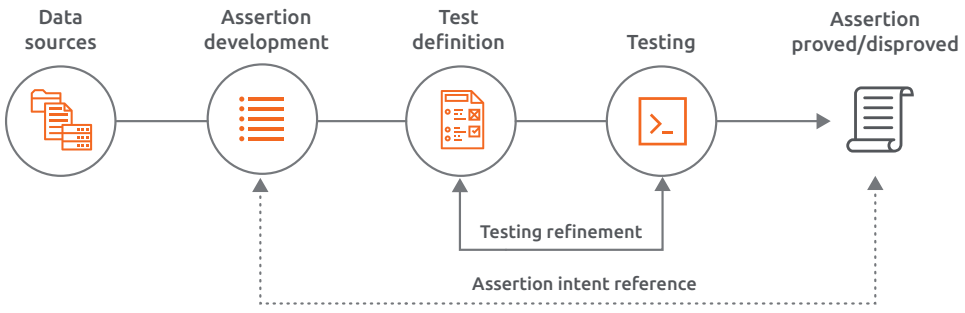

Figure 4. Evaluation methodology

Summary of observations

After testing, a detailed list of observed potential risks and technical recommendations was developed and shared with Canonical. Given the large scope of testing, with multiple review domains, the number of observations was relatively low. Though recommendations were identified that present opportunities for evolution of the platform in some domains, the lack of high-risk findings supports Ubuntu Core as a reasonable choice to securely deploy and maintain applications on IoT devices and other embedded systems.

Ubuntu Core Strengths

The cybersecurity strengths of the platform are clear, but as is always the case, organizations should perform due diligence and investigate any platform before diving in. To that end, the findings table below provides insights into platform strengths and key capabilities.

Domain	Strengths
Brand Snap Store	Access controls and authorization decisions are sound and incorporate elements and controls that would be expected for similar service platforms today.
	Cryptographic controls and signing are used in an appropriate way to help manage snap integrity and grant assurances to consumers.
	Device binding to brand stores helps reduce the risk to deployed systems and devices of unauthorized snap installations or updates from low-trust sources beyond those approved by the brand.
	Careful consideration has been afforded to layers and models of trust within the brand snap store, enabling adaptability in line with the adopter-specific security needs.
Public Snap Control/ Protections	Sandboxing protections and a combination of automatic review and trigger-based human review provide layered defenses against snap compromise.
	Policy-based snap inspection and interface declaration assertions help ensure timely review while balancing agility and rapid deployment needs.
	Effective use of channels and confinement modes helps to ensure production risk is minimized while enabling a flexible development and testing experience.
	Publisher (adopter) trust remains a key aspect of the trust model to support highly variable use cases, implementation needs, and security requirements that are specific to the adopter — as opposed to mandating requirements.

Domain	Strengths
Snap Package/ Container	Snaps are controlled and operated in a manner that prevents unplanned or unauthorized access to privileged system APIs, portions of the OS, or non-application-specific user data.
	By default, snaps have significant restrictions that reduce risk to the underlying system, including restrictions related to user changes, job scheduling, unapproved hardware access, user management, security policy, kernel runtime variable, sensitive kernel syscalls, and others.
	The OS snap provides control enforcement as subsequent snaps are deployed, ensuring a more consistent and dependable configuration.
	Sensitive interfaces that provide privileged access and are configured for auto-connection are blocked and carefully vetted, requiring manual review.
	Strict confinement with manual interface control can be used to implement fine-grained snap protections in higher-risk scenarios.
Snapd and Snap Deployment	As a shared strength and also potential point of awareness, snaps are based heavily on debs and are subject to the vetting and validation method used by them in updates/releases.
	Snaps provide a means for safe and effective rollbacks in the event of bugs or deployment issues.
	Snaps and sandboxing help isolate the broader system from various failure scenarios that may otherwise have broader reach on traditional systems.
Trust Models/ Ecosystem	As an open source platform, there are structures in place that provide public visibility and feedback mechanisms into Canonical's mediation of issues and changes.
	Reasonable and expected effort is applied to establishing trust through cryptographic means.
Ubuntu Core (Kernel + Utilities)	The Ubuntu Core OS snap provides a relatively lightweight and known profile (built from trusted debs) upon which to layer trust and additional points of control mediation. The base system itself contains little more than the kernel, the init process, snapd (a separate snap in later versions), standard Linux/UNIX tools, libraries to support these tools, and a limited number of common tools to support application development.
	A minimized attack surface relative to traditional deployments is used to help reduce risk to systems.
	Significant settings and configuration artifacts beyond default services and exposure profile (e.g., provisioned user logins, well-known system accounts, SSH password-based logins) are configured in a secure, restrictive manner out of the box.
	Sandboxing and snap confinement are key tenets of the Ubuntu Core approach to risk reduction. Application confinement is enforced during runtime via discretionary access controls (DAC), mandatory access controls (MAC), AppArmor, seccomp kernel system call filtering (limiting the system calls a process may use), and cgroups device access controls (for hardware assignment).

Operational security considerations

Deploying any system into an operational environment requires careful thought and planning; even the best-architected, highly secure platform can be rendered insecure if integrated incorrectly or without proper regard for its intended controls and design elements.

The Ubuntu Core ecosystem is designed to provide a scalable, secure solution for large fleet development and operations. While the specific requirements of every organization, application, or team are different, there are many components of Ubuntu Core that likely interact with operational security and should be carefully considered as part of a production deployment.

During this analysis of Ubuntu Core, threat modeling activities identified the following potential operational security interactions. The development and operations teams responsible for production deployment of Ubuntu Core may find these useful to consider.

> "Even the best-architected, highly secure platform can be rendered insecure if integrated incorrectly or without proper regard for its intended controls and design elements."

Domain	Opportunity	Mitigating Factors, Controls, or Notes
Brand Snap Store	Key creation and key signing role constraint improvements should be made to help manage deployment risk and bind specific operations to appropriate key classes.	Platform adopter, developer, or operator key management controls can reduce the risk of improper use.
	Require multifactor authentication for snap store SSO logins as the default. Roll out activation and enrollment such that Canonical employee intervention is not required.	Platform adopter must expend the effort necessary to enable. Ensure trust model and delegation to Canonical is understood.
	The brand organization (as opposed to Canonical) should own account lifecycle management.	Periodic audit and review of accounts by platform adopter to ensure no unauthorized changes are made.
	Improve documentation accuracy and completeness related to permissions, accounts, and overall brand store process.	Mainly an adoption-issue, and ensuring behaviors align with expectations.
Snap Package/ Container	Network traffic constraints should be considered and improved relative to snap requirements to mitigate the risk of what is today essentially a binary decision (access or no access).	Egress firewalling, appropriate network placement, or other similar controls added by platform adopter can reduce the risk of unauthorized egress traffic.
	Though more supportive of legacy software, strict snaps would benefit from running as a non-root user by default.	Platform adopter awareness of snap context, and appropriate vetting and trust management related to snaps.

Domain	Opportunity	Mitigating Factors, Controls, or Notes
	AppArmor policy improvements would help prevent snaps from harvesting information beyond ideal boundaries as is possible today even without the system-observe interface enabled (e.g., "ps aux" output may contain data that would be ideal to restrain).	Platform adopter awareness regarding cross-snap visibility and potential leakage vectors.
Trust Models/ Ecosystem	The voting system for interface auto-connection could improve accountability by integrating technical peer review requirements and transparency in the form of an immutable ledger of review and approval.	Platform adopter awareness and due diligence regarding snap configuration defaults (similar to user responsibilities regarding software library risks).
	There should be no option to create and use passphraseless keys, and reasonable passphrase requirements should be enforced.	Platform users can ensure administratively that no such keys are created.
	The trusted root CAs included, while useful and ensuring improved as-built compatibility, are too numerous and should ideally require explicit addition as part of the configuration process to minimize the risk related to CA compromise and abuse.	Platform adopter may modify certificates to meet their needs and trust requirements.
Ubuntu Core (Kernel + Utilities)	Read-only root filesystem control points, while foundationally strong, can be overridden with access to a root shell. It is important this be clearly conveyed, and the importance of root access controls reinforced.	Platform implementer must ensure strict control regarding access to root shells in all cases in order to ensure the control point is intact. This mitigating factor should always be an objective.
	SSH is enabled by default, and though valuable in some scenarios, it is likely to be undesirable in many. Ideally, default configurations would support the most secure configuration with options to provide added functionality if needed. Options such as limited time-bound service availability may be reasonable as a compromise. Additionally, default accepted ciphers are too generous (e.g., HMAC-SHA1).	Gadget settings can be used to override the Canonical default selection where appropriate. Adopters should be aware of the default state. Deployers could (and should when appropriate) adjust and strengthen ciphers to meet their specific needs.
	Passwordless sudo misses an opportunity to provide an additional layer of protection following access to a management shell.	Adopters can integrate changes to this default behavior into build and CI/CD procedures to improve the default state.

The Case for Adoption of Ubuntu Core

Increased competition in the IoT, medical, and robotic device space has raised the pressure on manufacturers and developers to rush products to market. Historically, this rush to market has resulted in lack of adequate thought, planning, and tools around device security. Often, support for software and OS updates, and other long-term lifecycle management activities, is inadequate or nonexistent.

Traditional embedded system models have left much to be desired. One approach has been to develop an all-custom platform or utilize a commercial microkernel. In the era of highly networked and integrated devices, this approach has become impractical, simply due to the amount of base functionality that must be implemented to be part of a larger networked environment. This includes the need for a network stack, logging, patch management, event monitoring and correlation, cryptography, integration APIs, centralized authentication, and shared database access, to name a few.

On the other side is the approach of building upon an existing minimalist Linux distribution. This has the advantage of a ready-made team of qualified developers and a rich set of available functionality, but is missing basic security structures, including:

- application validation and sandboxing;
- fleet management facilities to provide OS, module, and application updates; and
- CI/CD pipeline management.

In short, the "Let's just build this on Linux" approach might be great for a university research project or single-instance proof-of-concept deployment, but it fails spectacularly at any level of enterprise or customer scale, especially over time.

Ubuntu Core is the thoughtfully architected middle ground, and the case for its adoption boils down to four key points.

All the power of Linux

Ubuntu Core harnesses all the power of Linux and the already abundant and growing Ubuntu snap community. This creates a conduit to the latest and greatest features, device support, third-party platform integrations, and standard interfaces such that a developer can focus on their unique value proposition and not waste time writing code to solve a problem that's already been solved.

Structured security

The Ubuntu Core ecosystem provides a complete package for *structured, maintainable* security. While there is no zero-risk answer when deploying networked devices, as validated as part of the testing performed for this white paper, Ubuntu Core presents a reasonable balance of end-to-end lifecycle security controls within a framework that empowers the developer with a wide array of flexibility. The structure of the ecosystem is such that it enables application of fine-grained security controls, but in a manner that reduces the overall long-term support workload.

Velocity

The embedded systems/IoT world is fast-paced, with no margin for error on timing. The snap-based approach of Ubuntu Core enables rapid development and deployment of initial functionality, as well as long-term functionality add-ons and enhancements, all built into the ecosystem toolset. Further, snapd roll-back capabilities help reduce velocity risk and provide rapid recovery paths.

Secure lifecycle management

Ubuntu Core also solves the device registration, management, and update problem, including closing the loop for ensuring that updates are authentic. Security is only as strong as the weakest link, and in that light, it is imperative that every layer of the device be easily patchable, with

integrated code authenticity validation such that unauthorized or malicious code cannot be introduced. This whole process is backed by Canonical's commitment to provide 10 years of OS-layer patches for the platform, ensuring each device can be properly maintained throughout its usable life.

Summary and Conclusion

Ubuntu Core provides a comprehensive, secure ecosystem that solves many of the challenges associated with traditional Linux distribution models while providing developers with flexibility and control.

While there are many positive attributes of the Ubuntu Core approach, perhaps the most attractive is the security-first approach that provides *structured, maintainable* security in every aspect of the system and the device lifecycle. This attribute was the primary focus of the testing performed within the scope of this white paper, as trust in security is only as deep as the independent review that has been performed.

The data-driven approach used for this security testing ensured that all aspects of the system were evaluated and included everything from user workflow and interfaces, to brand snap store functions, to application security policy and sandboxing restriction effectiveness, to the deployment and update process.

At the highest level, no critical findings were identified through testing. This validates Ubuntu Core as a secure, well-balanced platform choice for deploying applications on embedded systems for IoT devices and other similar large-scale deployments.

It's true that there is no perfect approach to secure ecosystem design and management, but Ubuntu Core represents a significant step forward in a holistic approach to this problem – it brings all of the power of the Linux and snap world to the developer's fingertips, while providing just enough structure and power through fine-grained security controls, hardening, and sandboxing in an open source platform that provides for long-term fleet lifecycle management. These attributes together form a security arbitrage that is a win-win for the IoT world.

> "Ubuntu Core provides a comprehensive, secure ecosystem that solves many of the challenges associated with traditional Linux distribution models while providing developers with flexibility and control."

Appendix – Security Control Point Sources

- **Ubuntu Core Security Whitepaper** [version 3.0.0]
  https://assets.ubuntu.com/v1/66fcd858-canonical-ubuntu-core-security-2018-11-13.pdf
- **Security and Sandboxing Documentation**
  https://docs.ubuntu.com/core/en/guides/intro/security
- **Ubuntu Core Security Policies**
  https://docs.ubuntu.com/core/en/guides/intro/security#working-with-security-policies
- **Snapcraft Documentation – Confinement**
  https://docs.snapcraft.io/snap-confinement/6233
- **Snapcraft Documentation – Kernel Snap**
  https://docs.snapcraft.io/the-kernel-snap/697
- **Snap Security Overview** [official blog]
  https://snapcraft.io/blog/where-eagles-snap-snap-security-overview
- **Tools for Making the Snap Trek Easier** [official blog]
  https://snapcraft.io/blog/snap-up-your-development-tools-for-making-the-snap-trek-easier

Rule4, Inc. / 3002 Bluff Street, Suite 100, Boulder, CO 80301 / +1-720-580-5888 / www.rule4.com

Suggested Citation – Rule4. (2020). *Ubuntu Core Cybersecurity Analysis* [White paper].