Canonical

WHITEPAPER
Q1 2025

Security in depth with Ubuntu

Mapping security primitives to attacker capabilities

Contents

Executive summary

Ubuntu's security strategy is a multi-layered defense system, designed to counter specific threats with calculated resilience. Each security feature works in harmony to create a robust platform capable of withstanding sophisticated attacks. By recognizing the unique threats each layer addresses, you can choose the defenses best suited to your environment.

Extended Security Maintenance (ESM) lays the groundwork for this security response, swiftly addressing known vulnerabilities with up to 12 years of dedicated patching. With AppArmor in place, zero-day threats are caught in tailored confines, allowing applications only the access they need and nothing more, thwarting attackers' lateral moves within systems.

Ubuntu's alignment with FIPS standards ensures cryptographic integrity, essential for regulated industries, while CIS benchmarks deliver customizable system hardening that balances security and flexibility.

For those aiming to infiltrate at first boot, Secure Boot and Full Disk Encryption (FDE) stand as gatekeepers. Secure Boot enforces a chain of trust, letting only verified code through, while FDE encrypts data at rest, rendering it untouchable without the encryption key, useless to any intruder.

For the most formidable adversaries, Confidential Virtual Machines (CVMs) reshape the rules. With Intel TDX and AMD SEV SNP, Ubuntu builds a secure haven within the CPU itself, decoupling data access from resource management. Even a compromised hypervisor or malicious insider finds no entry to sensitive data, only impenetrable isolation.

Ubuntu's layered approach offers a symphony of security, measured, deliberate, and adaptable, ready to meet today's threats and tomorrow's unknowns with steady resilience.

Introduction

Back in the 1940s, the cybersecurity community flirted with the idea of perfect security through the one-time pad. Imagine: a cipher with a key as long as the message, used only once, and with true randomness to guarantee absolute secrecy. It was airtight and theoretically unbreakable. However, perfection has a price: generating random keys and distributing them securely became a logistical nightmare, and reusing keys carried a risk of leaks. So, instead of perfect security, we shifted toward computational security, a realistic practical model that doesn't seek absolute unbreakability, but rather aims to make attacks so hard they're practically impossible.

Consider AES-256 encryption. Its security doesn't come from a guarantee that no one could ever break it. Instead, it banks on the assumption that no attacker today (short of some dystopian quantum supercomputer) could feasibly brute-force the key. AES-256 stands firm not because it's unbreakable in theory, but because it would take millions of years for an attacker to crack it. It is secure precisely because of what we assume about the attacker's computational limitations.

Public key cryptography takes this concept and expands it. When we use algorithms like Rivest–Shamir–Adleman (RSA) or Elliptic Curve Cryptography (ECC), we're banking on the assumption that certain mathematical problems, like factoring large numbers or solving discrete logarithms, are computationally infeasible with current technology. The beauty (and the gamble) of public key cryptography rests on a crucial, open question in computer science: is Polynomial time (P) equal to Nondeterministic Polynomial time (NP)?

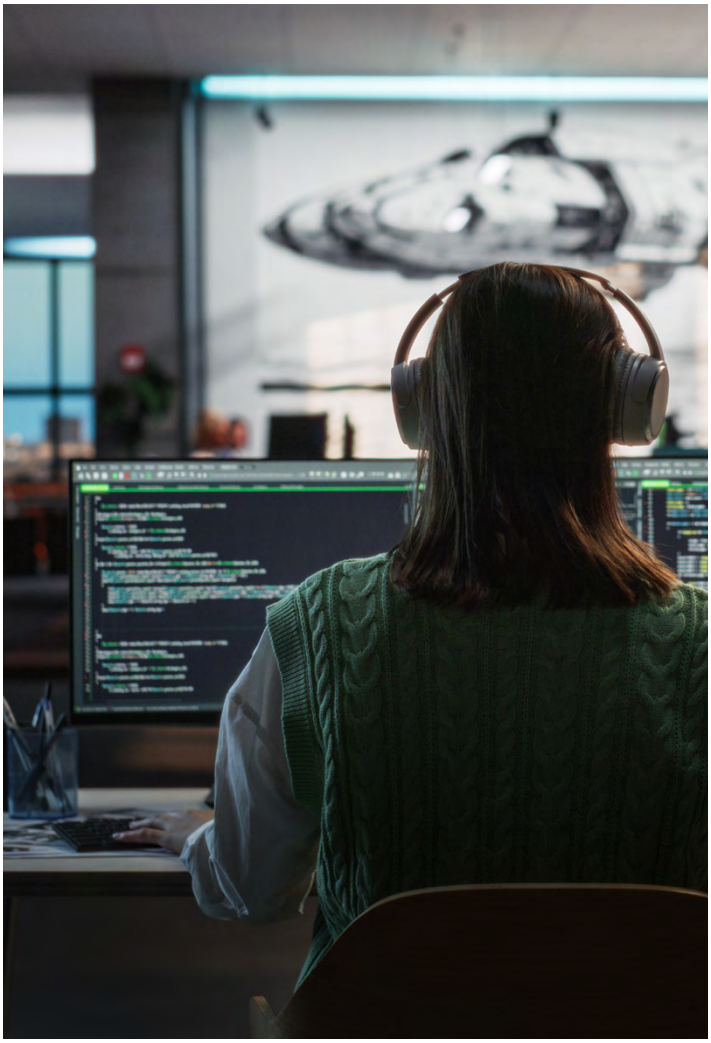
This question, unsolved and fundamental, is a cornerstone of our security architecture. If P were to equal NP, our whole system could crumble, because it would imply that what we think is hard, like breaking RSA keys, might in fact be easy with the right algorithm.

Indeed, we are all playing a game in cybersecurity: a delicate, high-stakes game defined by who we think is waiting on the other side of our defenses. Forget about building walls that no one could ever breach; instead, we're aiming to build fortresses that hold up against the resources we expect attackers to bring to the table.

As such, our strongest defense is a security-in-depth approach, where multiple, independent layers of security work together to protect every level of a system. Instead of relying on a single line of defense, each layer, whether it's network protocols, access controls, encryption, or physical barriers, serves as an additional barrier to delay, detect, or thwart attackers. This strategy embraces the reality that no individual layer is infallible, and instead focuses on building resilience through redundancy and diversity, making it exponentially harder for an adversary to compromise the entire system. Security in depth is adaptive, anticipating evolving threats across the cybersecurity landscape, and as such, it remains one of the most effective frameworks for comprehensive protection.

In this paper, we'll journey through the landscape of the security primitives and solutions that make up Ubuntu's security in depth strategy, all through the lens of computational security. From Extended Security Maintenance (ESM) all the way to confidential computing. For each section, we'll analyze the specific threats it addresses, the attacker capabilities it assumes and how these solutions work together to create a layered, defense-in-depth approach. Think of it as an upgrade path, from defending against the simplest threats to countering sophisticated adversaries.

Defend against known vulnerabilities



At the most basic level, security begins by defending against attackers who exploit known vulnerabilities. The moment a vulnerability is made public, it's not just your IT team that knows, attackers around the world have the same information. Once a CVE hits, exploits are quick to follow, often spreading in days or even hours. And when those exploits land in automated attack kits, you're racing against time: every hour without a patch is an hour where your system could be compromised.

Unfortunately, organizations need 97.8 days on average to fix a vulnerability, according to Snyk, and in a report published by Verizon 2022, only 25% of the scanned organizations were found to patch known vulnerabilities within two months of their public disclosure. In today's modern cyber risk landscape, this has to change. Entry-level approaches to cyber security need to closely monitor newly discovered vulnerabilities and automate patching for these CVEs as soon as they are discovered. This is especially true where your operations' environment spans multiple platforms: servers, SDKs, tech stacks, and mixes of open source and proprietary software.

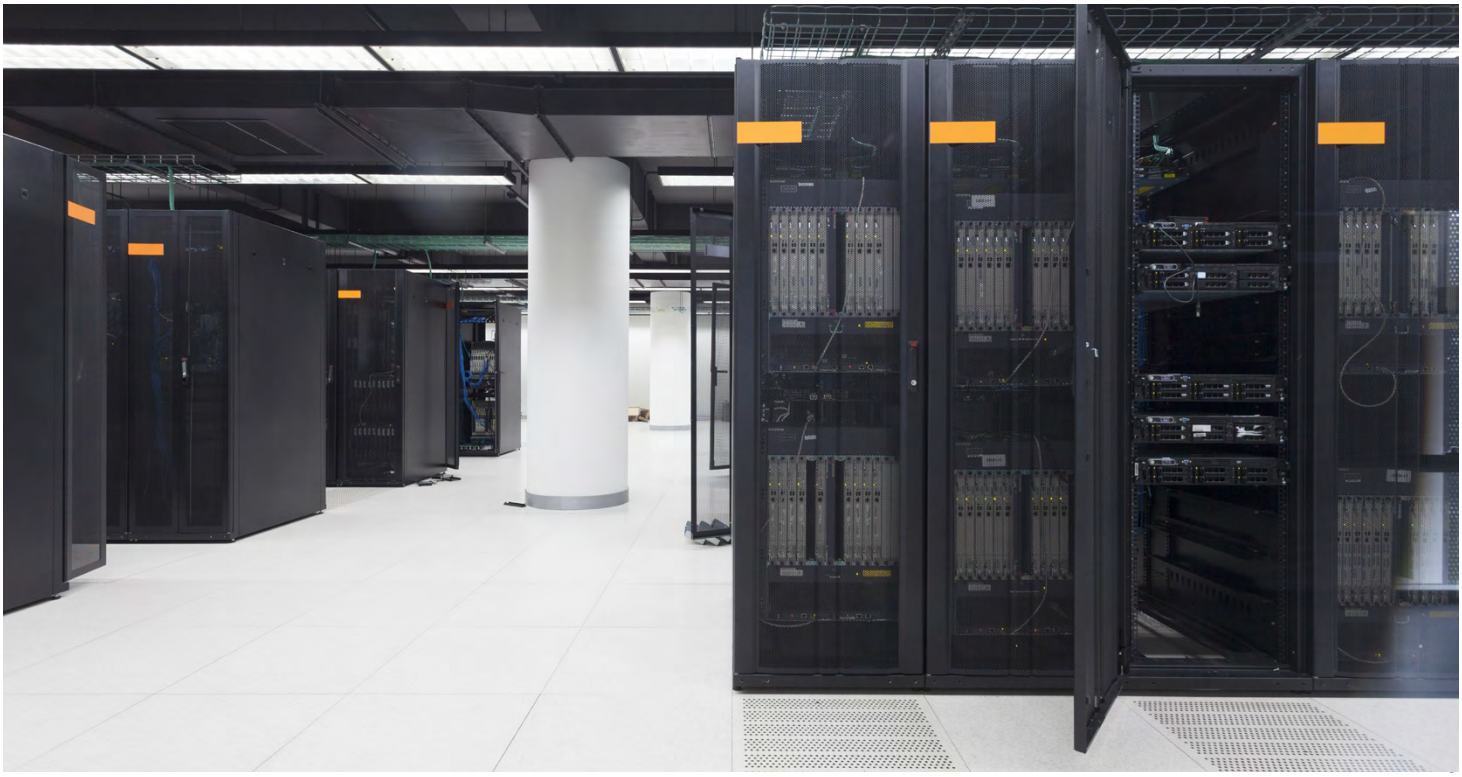
For the Ubuntu ecosystem, Each LTS release of Ubuntu benefits from up to 12 years of security updates with an Ubuntu Pro subscription. Throughout this period, the Ubuntu security team takes in vulnerability reports every day from MITRE, NVD, and other sources to continuously develop and publish fixes as soon as security issues are discovered, often before vulnerabilities are even made public. This security patching covers the open source packages that form the base of Ubuntu Main repository (2,300 packages), as well as over 23,000+ packages in the Ubuntu Universe repository, which include web servers, databases and development tools. Together, they form a single trusted secure repository that covers all the open source software required by users. With Ubuntu Pro, the currently measured average time to fix for Critical CVE vulnerabilities is less than 24 hours.

Strengthening cryptographic integrity implementation

Now, let's imagine an attacker who's a bit more advanced: one who would try to exploit flaws in cryptographic algorithms or their implementations. Such an attacker may not crack your encryption outright but may use techniques like padding oracle attacks on improperly implemented encryption schemes, or timing attacks to infer information by analyzing how long certain cryptographic operations take. Side-channel attacks like these exploit subtle flaws in implementation rather than brute-forcing the encryption. For instance, an attacker might target a system using AES-CBC (Cipher Block Chaining) mode without proper padding, enabling a padding oracle attack that could gradually reveal plaintext information.

Alternatively, an insecure implementation of RSA could leak private key bits through timing variations, allowing an attacker to reconstruct the key over multiple attempts.

To mitigate these risks, it is crucial that enterprises avoid applications that embed unvalidated cryptographic modules, or use cryptographic libraries in ways that do not conform to their security policies. For environments with higher compliance needs, especially in government and regulated industries, Ubuntu offers FIPS (Federal Information Processing Standards) compliance. FIPS-certified modules in Ubuntu mean that all cryptographic functions, from hashing to encryption, meet the strict standards set by the National Institute of Standards and Technology (NIST). Ubuntu supports FIPS on Linux with a series of validated components: the Linux Kernel Crypto API, OpenSSL and OpenSSH, libgcrypt, and strongSwan.



Guarding against misconfigurations

Now, consider a more resourceful attacker. Even with a perfectly patched software stack, this attacker will target systems with configuration weaknesses, settings left open or poorly secured, whether through default configurations or human error. Such an attacker is capable of probing for weaknesses in system setup rather than software code. To prevent attacks by this actor, it is vital to ensure robust and secure configurations in your system.

While the default configuration of Ubuntu LTS releases already balances between usability, performance and security, mission-critical systems can be further hardened to reduce their attack surface. Reducing the attack surface is a widely accepted security best practice, and is often required by cybersecurity frameworks.

Canonical works with industry leading organizations, such as CIS and DISA, to produce security hardening benchmarks for Ubuntu. These security benchmarks contain hundreds of steps which can be prohibitively time-consuming to apply manually, so we provide the Ubuntu Security Guide (USG) – a tool based on OpenSCAP – to automate the process. USG can generate remediation scripts to harden a system in one procedure, as well as producing audit reports detailing the hardening rules that have been applied. USG profiles are available for CIS benchmarks and DISA STIGs.

Some of the hardening steps that are made available to Ubuntu systems are:

- Disabling unused USB ports. If the server doesn't need to use certain hardware features, these can often be disabled in the BIOS, to further prevent physical attacks against the system.
- Configuring the disks with full disk encryption. Disk encryption means that someone can't take the disks out of the machine and access or modify the contents offline. In cloud environments and virtual machines, disk encryption also prevents a malicious actor at the hypervisor level from accessing your virtual disks.
- Removing unnecessary and unused components. A Linux OS such as Ubuntu contains packages to cover a very wide range of use cases, but it's likely that a production system will only have a small number of critical workloads. Any package not supporting these workloads should be removed.
- Tightening default settings and enforcing encryption. Ensure directories are configured to allow only the minimum privileges required to run the production workloads, and encrypt file systems.
- Configuring logging and integrity checks. All system and application logs should be stored on a remote server to ensure that in the event of a hack the attacker can't delete the logs to cover their tracks. File integrity monitoring software should be deployed to provide warnings if any unexpected changes occur which might be indicative of an attack.

Hardening benchmarks have broad applicability across a wide range of industries, and are useful for any organization deploying services on the internet. Some industry sectors carry specific regulatory requirements which mandate system hardening, such as the Payment Card Industry Data Security Standard (PCI-DSS).

Protect against zero day vulnerabilities with Apparmor

Now consider an even more sophisticated attacker, armed with a zero-day vulnerability and poised to exploit a critical flaw in an application's code. Their goal: gain unauthorized access, execute malicious actions, or move laterally within the system.

Clearly, there is no patch to apply here, and no patching solution will serve as an effective defense until the vulnerability is disclosed. However, if AppArmor is in place, the impact of a potential exploit can be significantly constrained. AppArmor enforces Mandatory Access Control (MAC) through profiles that define strict limits on what applications can access and do, effectively containing potential exploit pathways, even for unknown vulnerabilities.

Ubuntu comes pre-installed with a range of AppArmor profiles for common applications. Furthermore, if your critical workload application doesn't have a profile, it is straightforward to create one. This is particularly important for any network-facing processes. In order to generate a new profile, AppArmor can be put into Complain-mode while the application is run with its full range of capabilities, capturing the actions taken into a file. This file can then be used as a new profile for Enforce-mode.

Securing your early boot software

Now let's imagine an attacker targeting your system even before it fully starts, aiming to insert malware during the earliest stages of booting up. This is a sophisticated and stealthy approach, as early boot software and firmware are foundational components, responsible for initializing the hardware, verifying components, and loading the operating system (OS) itself. By compromising this phase, attackers can establish a foothold that is deeply embedded, difficult to detect, and nearly impossible to remove without specialized tools.

Early boot software and firmware, including the bootloader, UEFI/BIOS firmware, and trusted computing base, are critical because they act as a root of trust for the entire device. When these components are tampered with, they allow malware to execute with the highest privileges, even before the OS security mechanisms are loaded.

One of the major challenges in protecting early boot firmware is that it often resides on non-volatile memory (e.g., flash memory) on the motherboard. This memory can be modified if not properly protected, allowing attackers to inject malicious code that will persist even after reboots or factory resets. Firmware-based malware can hide itself by avoiding detection mechanisms that operate at the OS level, making it invisible to most security solutions.

Given these risks, securing early boot software and firmware is essential. Techniques like Secure Boot, which ensures only trusted, signed firmware is executed, and hardware-based protections, like TPMs or Platform Security Processors (PSPs), can help mitigate these risks. Furthermore, firmware updates should be cryptographically signed and authenticated to prevent unauthorized modifications. Detecting and protecting against these threats requires a robust, multi-layered security approach that spans hardware, firmware, and software, ensuring each stage is safeguarded to maintain the system's integrity from power-on through to full operation.

Ubuntu defends against these high-level threats with two critical protections: Full Disk Encryption (FDE) and Secure Boot.

- Secure Boot: it enforces a chain of trust during the boot process, preventing unauthorized code from running and ensuring that only signed, trusted software is loaded. In Ubuntu's Secure Boot process, all pre-built boot binaries, except the initial ramdisk (initrd) image, are signed with Canonical's UEFI certificate, embedded within the shim loader signed by Microsoft.

- Full Disk Encryption (FDE): it ensures that all data on the disk is inaccessible without the encryption key, safeguarding the device even if an attacker gains physical access. Enabled through the Logical Volume Manager (LVM) and Linux Unified Key Setup (LUKS) during Ubuntu installation. This means that unless the correct decryption password is provided, it's impossible for an attacker to read, write, or alter any files on the disk.

Defending against a compromised host with confidential computing

Finally, let's picture the strongest type of adversary: one capable of compromising the host environment itself. This attacker can infiltrate the platform's firmware and privileged system software, including the host OS and hypervisor, and may even have the backing of malicious administrators.

Traditionally, this level of access would expose your workloads to significant risk, with any vulnerability in the privileged system software compromising the confidentiality and integrity of your data and code. Historically, the hypervisor, host OS, firmware and DMA-capable devices were implicitly trusted to access and manage virtual machine (VM) resources. Since these components managed execution, memory and hardware access, they were granted access to workload data – a necessary risk for operational control. However, this model left sensitive code and data vulnerable to host-level threats.

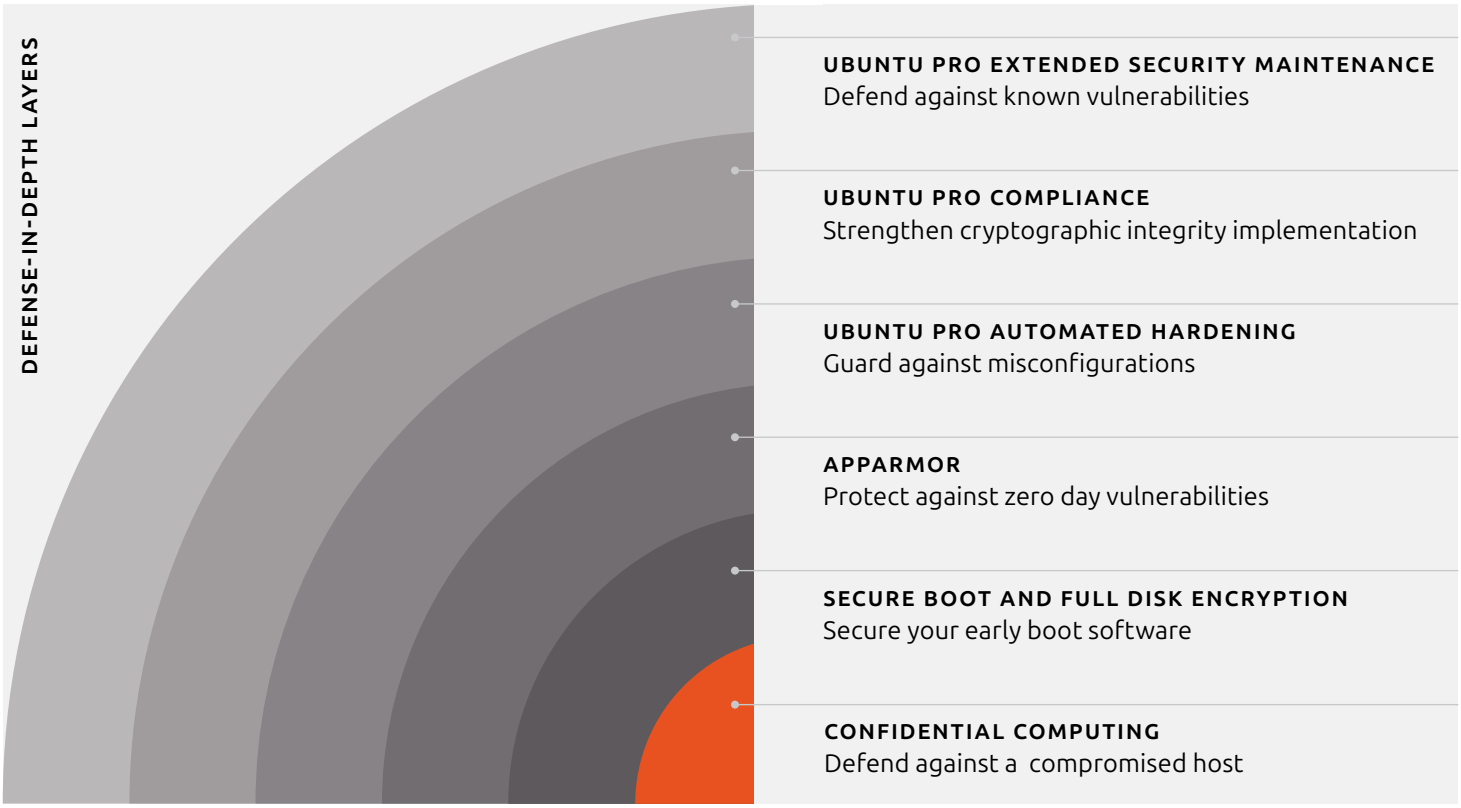
Today, confidential computing has emerged to fundamentally shift this paradigm by decoupling resource management from data access. Through this new security primitive, privileged system software like the hypervisor can still manage VM resources without direct access to the data within the VMs. This means that, even if a vulnerability exists in the hypervisor or host OS, it cannot compromise the security of confidential workloads.

To achieve this level of isolation, new CPU security extensions such as AMD SEV (Secure Encrypted Virtualization) and Intel TDX (Trusted Domain Extensions) have been developed, introducing two critical security pillars:

1. **Memory Isolation through Hardware-Level Encryption:** CPUs with confidential computing capabilities use an AES encryption engine within their memory controller to encrypt and decrypt memory pages with each read or write operation. This ensures that workload data is stored in encrypted form in system memory, accessible only within the secure execution environment. The encryption and decryption happen transparently within the CPU, providing strong memory isolation without performance penalties.

2. **Hardware-Based Access Control:** Confidential computing-enabled CPUs incorporate new instructions and data structures that allow auditing and control over memory management and platform device access, typically handled by privileged system software. These mechanisms help detect unauthorized modifications and replay attacks by tracking the memory pages mapped to confidential workloads, preserving data integrity against tampering.

To help users harness these hardware-rooted security guarantees, Canonical offers Ubuntu Confidential Virtual Machines (CVMs) powered by Intel TDX and AMD SEV SNP. Ubuntu CVMs create a hardware-rooted execution environment that logically isolates workloads from the host, reducing the trusted computing base to just the application and the CPU. Ubuntu CVMs are available today on all major public cloud providers, and if you are a private cloud customer, you can also use Ubuntu on both the host and the guest to build your confidential private cloud.



Conclusion

Every security measure Ubuntu designs is crafted to withstand the capabilities of a hypothetical attacker, smart enough to pose a threat, but not infinitely resourced. It's a calculated balance, one that frames everything we do in cybersecurity, from encryption choices to intrusion detection. This is where we live: in a world where "good enough" is not an admission of defeat; it is our best strategy, in a chess game against the unknown.

As such, Ubuntu's security offerings are much more than just a collection of tools. They are an ecosystem of layered defenses, each tuned to address specific threat levels and attacker capabilities. By understanding the unique threats each measure counters, you can make informed choices about which defenses are most important for your environment.

From patching known vulnerabilities with ESM, to hardening configurations, securing the boot process, and protecting against malicious host environments, Ubuntu gives you the flexibility to adapt to a range of security needs. This layered approach, with clear mappings from defense to threat, isn't just practical; it is powerful. Whether you're managing a small team or securing critical infrastructure, Ubuntu's depth of security provides the resilience and confidence to meet both current and future threats head-on.

Learn more about Ubuntu Pro and start building your secure infrastructure now.

Our security experts are also ready and glad to discuss your security needs in more detail.

Reach out today to discuss how we can secure your environment.

ADDITIONAL RESOURCES

- Ubuntu Pro
- What is System Hardening? Essential Checklists from OS to Applications | Ubuntu

Canonical Limited

Registered in Isle of Man,
Company number 110334C

Registered Office

2nd Floor Clarendon House
Victoria Street, Douglas

IM1 2LN
Isle of Man

Canonical