Smarter and More Secure Development with SUSE® Embedded Solutions
The Security Advantages of Commercial Linux for Embedded Solution Development

Security awareness has increased across all computing platforms as embedded solution developers strive to build interoperable, connected devices, hardware, and systems.

One of the first major decisions they face is: which operating system base should they develop on, and support? This question is closely followed by: how will security be managed? Embedded solution developers are placing a renewed focus on how the underlying operating systems selected for development will enable them to mitigate and easily manage security risks.

Section 1: Introduction/Executive Summary

Linux has changed the embedded systems landscape, and is quickly becoming the preferred embedded solution development platform. In fact, embedded Linux systems are growing at a CAGR of 16.7 percent for open Linux, says VDC.\(^1\)

In this white paper, aimed at embedded solution developers of purpose-built appliances, devices, gateways, and systems, we will consider why the underlying operating system is critical to achieve security. We will then describe the common security challenges encountered when building and maintaining embedded devices and hardware. This discussion leads into why Linux—and, in particular, a SUSE Embedded Solution—is the preferred operating system choice for secure embedded development. Next, we will provide best practice examples of companies that are bringing SUSE Linux Embedded Solutions to market quickly and securely. Finally, we will suggest ways embedded solution developers can get started with a SUSE Embedded Solution to minimize costs and simplify security in their development efforts for faster time to market.

Section 2: Security Concerns Rise in Embedded Solution Development

While in the not-so-distant past many devices were isolated point solutions, the current trend towards interoperability and connectivity of device functions—often referred to as the Internet of things (IOT)—is raising concerns about the security of embedded devices and systems. Since these connected devices are used in critical infrastructure, or impact the health and safety of individuals, security is of paramount importance.

To that end, embedded solution developers are seeking ways to implement more stringent security protocols to protect the functionality, data, and well-being of both their customers and operators. Developers are under significant pressure to support new features and bring solutions to market faster and more efficiently. To be successful, they need to develop secure hardware and systems that can be maintained over the life of the product, while keeping development costs down. As more functionality is embedded into smaller device footprints, security concerns, and the ongoing ability of device manufacturers to effectively manage these devices, becomes more complex.

In an embedded development environment, patches and updates are not always easy to administer because not all devices are connected to the Internet. Some updates may even require physical access to the device—which is not always practical. Bruce Schneier, security guru and blogger, was quoted saying, “the Internet of Things is wildly insecure—and often unpatchable.”

Common coding mistakes can, and often do, happen in software implementations, accounting for a large number of security risks. In fact, according to CERT, 64 percent of the vulnerabilities in its National Vulnerability Database were caused by programming errors. These critical errors can be exploited and expose customers and operators to compromised functionality—sometimes with devastating effects to customer trust, brand reputation, revenue loss, and even legal ramifications.

“We live in an ever-more connected world, and the easy answer of isolation is not viable... with sound, security-based and tried-and-true architectures that also take into account the time-value of the data/ assets being protected, it is possible to address cyber-threats in a rational, yet not overly complex or expensive fashion. Using diverse levels of protection is the best way to prevent problems.”

SOURCE: GARTNER

To maintain secure devices and mitigate their overall risks, embedded solution developers require seamless patch and update capabilities. Effective updates address software bugs and flaws that can alleviate vulnerabilities. Without updates and patches, embedded systems may be more easily compromised.

**Section 3: Security Considerations for Embedded Solution Development**

From the outset, security needs to be a key ingredient in embedded solution development—not an afterthought.

Software security is a complex challenge as software has many of its own security features, such as authentication methods, encryption, intrusion prevention and detection, and backup. At the same time, it can also contain errors (both deliberate and accidental) that can affect the system’s security, including design flaws, programming errors, and backdoors.

Embedded solution developers need to ask targeted questions about the critical nature and manageability of security prior to selecting an operating system:

- How will the maintenance of security and patches be managed?
- What resources are available to maintain and monitor available security patches?
- Are enough resources available to maintain security updates/patches changes to the operating system?
- How frequently will patches and updates be issued?
- How many resources/personnel are needed to execute any security changes/updates?
- Who needs access to what resources or data?
- What is the required functionality for the solution?

---

1. www.schneier.com/about.html
2. www.medicaldesignandoutsourcing.com/designing-software-security-into-linux-based-medtech/#_
3. www.cert.org/secure-coding/products-services/secure-coding.html?
In addition to these questions, a seminal work on the core information security principles can guide design choices, implementation, and overall life cycle concerns of an embedded device. See Exhibit 1 below for key references to security considerations for embedded systems.

<table>
<thead>
<tr>
<th>Security Component</th>
<th>Brief Description of the Core Information Security Principles and the Key Security Questions</th>
<th>Linux Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize the attack surfaces</td>
<td>What are the minimum required components for a given solution? Helps during the testing and overall life cycle of the device or system by reducing the number of attack surfaces that will need to be secure and kept up to date.</td>
<td>Selectively install what is needed Configure what is installed Enable and protect integrity of the configuration Securely run what is enabled</td>
</tr>
<tr>
<td>Isolation/least privilege</td>
<td>What are the critical components that need to be shared? The guidance afforded by least common mechanism for information protection instructs a designer to tend towards isolation. For an architecture that avoids the sharing of critical components, this lessens the impact a compromise may have on overall security.</td>
<td>Multiple technology approaches to isolate components such as virtualization and containerization</td>
</tr>
<tr>
<td>Patch updating</td>
<td>How are updates going to be accommodated and what will the patch and update process look like? The single most important security feature of a solution is to be able to accommodate updates to address software bugs and flaws which may lead to vulnerabilities or compromises as described above.</td>
<td>Streamlined patch management and updates</td>
</tr>
<tr>
<td>Simplicity and usability</td>
<td>What functionality is required? Avoid complexity, or abide by the “economy of mechanism” mantra. Selectively include only that functionality needed in a given implementation. Extraneous or auxiliary functionality can be left out, simply by not installing it or its dependencies.</td>
<td>Packaged in a modular, granular format that enables you to install only what you need.</td>
</tr>
<tr>
<td>Controlled sharing</td>
<td>Who needs access to what resources or data? Controlled sharing focuses on who may access what resources, or data, in what mode. Minimizing access to the smallest set of users with the least permission necessary to accomplish the required task.</td>
<td>Rich set of access control list (ACL) technologies to limit process access to system resources.</td>
</tr>
<tr>
<td>Open design</td>
<td>What support do we have if things go wrong, or don’t work? Source code for the kernel and the user-space components, along with that for any changes, can be readily inspected.</td>
<td>Large groups of developers and testers are more likely to catch flaws. Transparency of the development processes to track issues, resolutions, and overall change management.</td>
</tr>
</tbody>
</table>

Exhibit 1. Key security considerations for embedded systems development and Linux strengths

Section 4: Security Advantages with Commercial Linux for Embedded Solution Development

The selection of an embedded operating system is a complex process that requires consideration of multiple factors that impact every aspect of the project—from development time to licensing costs. Embedded solution developers can elect to design, build, and support the operating system themselves, or procure it from an outside party. Competitive pressures to innovate are forcing embedded solution developers to dedicate their resources to value-added

---

tasks—designing systems that meet the specific needs of their target markets—not to building and maintaining an embedded operating system.

A primary driver behind the growth of Linux\(^\text{10}\) for embedded systems development is security (see Exhibit 2 below).

<table>
<thead>
<tr>
<th>(Percent of Respondents)</th>
<th>2008</th>
<th>2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house developed operating system</td>
<td>19.2%</td>
<td>12.2%</td>
</tr>
<tr>
<td>No formal operating system</td>
<td>35.3%</td>
<td>13.2%</td>
</tr>
<tr>
<td>Other (Commercial, open source, etc.)</td>
<td>45.5%</td>
<td>74.6%</td>
</tr>
</tbody>
</table>

Source: VDC | Research, 2016

Exhibit 2. Primary operating system use on current projects

As mentioned in Section 2, embedded solution development requires multiple updates and patches through the entire product lifecycle. If using a custom operating system, or unsupported Linux distribution, embedded solution developers need to include these security updates themselves. The financial commitment and number of man hours associated with developing and maintaining a custom Linux distribution or proprietary operating system can be prohibitive.

AuthO\(^\text{11}\) recently found that 85 percent of developers admitted that they had rushed applications to market despite having security concerns about the device. Embedded solution developers that use a commercial Linux operating system can be confident that security patches and updates are easily managed prior to launch, and over the course of a product lifecycle.

Commercial Linux distributions provide embedded solution developers with a number of security advantages over the “free”, or in-house developed operating systems including:

- Additional value-add development tools that assist with security
- Confidence that the operating system has been tested
- Regular product updates with maintenance and security patches
- Required security patches with streamlined delivery
- Access to training and support
- Software and hardware certifications
- Long-term support over the entire product lifecycle

Section 5: SUSE Embedded Solution: The Linux of Choice for Secure Embedded Systems Development

Not all commercial Linux distributions are created equal. Founded in 1992, SUSE is the world’s first provider of an Enterprise Linux distribution. SUSE is committed to delivering reliable, best-in-class security to its customers and to the Open Source community. SUSE believes that trust in Open Source Software security in general, and the user’s privacy in particular, is indispensable.

“Linux was designed as a multi-user operating system, with specific access control paradigms. Because it is by nature, open sourced, security defects can be observed, traced to a root cause, and fixed by any interested party.”\(^\text{12}\)

---


\(^{11}\) [https://auth0.com/blog/2015/11/06/surprised-turns-out-consumers-dont-trust-iot-security/](https://auth0.com/blog/2015/11/06/surprised-turns-out-consumers-dont-trust-iot-security/)

Why SUSE Embedded Solutions?

A SUSE Embedded Solution, based on the SUSE Linux Enterprise code base, offers a meaningful choice of components up and down the software stack to suit the needs of an embedded operating system. Including an enterprise grade Linux kernel, plus a core of available user-space packages, SUSE Linux Enterprise also provides a significant ecosystem of modules, extensions, and developer tools. Refer to Exhibit 1 to see Linux, and SUSE Embedded Solutions security strengths.

Embedded solution developers rely on SUSE Embedded Solutions to:

- **Speed time-to-market.** SUSE Linux Enterprise helps fixed-function device and server manufacturers bring innovative solutions to market quickly. SUSE’s mature, proven operating system and industry-leading support services help accelerate product design and development and help manufacturers reduce development costs.
- **Administer long life cycle management.** Many manufacturers lack the resources to monitor and update security and bug fixes and patch the operating system on an on-going basis for the life of a product. With SUSE Linux Embedded Solutions you get 13 years of life-cycle management.
- **Provide more flexibility.** Just Enough Operating System (JeOS) images and build descriptions are available to kick start embedded solution development creativity. With a SUSE Embedded Solution, developers can focus on creating innovative products that are secure and add identifiable business value. Exhibit 3 below highlights key capabilities of SUSE Embedded Solutions vs. other Linux Distributions.

<table>
<thead>
<tr>
<th>Tools and Processes</th>
<th>Often encapsulated by a defense in layers approach, SUSE Linux Embedded solutions afford:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Choice and implementation of many different technologies and tools to match the situation and requirements at hand.</td>
</tr>
<tr>
<td></td>
<td>Give developers the ability to partition or containerize risk and exposure—and consider the best way to present their application to the world.</td>
</tr>
<tr>
<td></td>
<td>Security tools at the network, physical and for start-up mechanisms to enable multiple levels of security.</td>
</tr>
<tr>
<td></td>
<td>Restrict access from a given program, application, or service to utilize various resources, like files or devices.</td>
</tr>
<tr>
<td></td>
<td>Control specific users limits to restrict or delegate authentication to various authoritative services, which can then be used to authorize specific access modes.</td>
</tr>
<tr>
<td></td>
<td>Read more about the tools available for SUSE Embedded Solution at: <a href="http://www.suse.com/partners/isv/developer_tools">www.suse.com/partners/isv/developer_tools</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Transparency of the development processes to track issues and identify resolutions. Well defined build, test, quality assurance, and distribution processes in place for the initial, released deliverables and security updates including:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Issue tracking systems are leveraged and utilized for overall change management of the included software and respective versions.</td>
</tr>
<tr>
<td></td>
<td>Automated, defined build process, in a clean software build environment based upon a minimal, targeted operating system version plus only the stated build dependencies.</td>
</tr>
<tr>
<td></td>
<td>Comprehensive testing performed including automated unit test cases, overall regression tests, and license and code coverage checks before becoming available for consumption.</td>
</tr>
<tr>
<td></td>
<td>View updates and details behind the why the changes were made for each package and version at: <a href="https://download.suse.com/patch/finder/">https://download.suse.com/patch/finder/</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support</th>
<th>SUSE Embedded Solutions offers 10 years of standard major-version product support plus three years of extended support.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SUSE’s dedicated team of security experts receives advance notification of security flaws and vulnerabilities, and works with the Open Source community to pro-actively fix those vulnerabilities prior to bulletin release to the public.</td>
</tr>
<tr>
<td></td>
<td>Fixes to the most critical security issues are available to the partner community nearly immediately, reducing exposure for the partner and the partner’s customers.</td>
</tr>
<tr>
<td></td>
<td>Learn more about SUSE Embedded Solutions support at: <a href="http://www.suse.com/support/">www.suse.com/support/</a></td>
</tr>
</tbody>
</table>

Exhibit 3. SUSE Embedded Solution security differentiators
Section 6: Conclusion/Additional Resources

Embedded solution developers are selecting commercial Linux operating systems over internally developed, or “free” Linux distributions to mitigate and easily manage security risks.

Commercial Linux distributions such as SUSE Embedded Solutions provide embedded solution developers with:

- Enhanced engineering and technical support services
- Comprehensive and easy-to-use development tools
- Access to training through the SUSE Partner Program
- Simplified subscription models and flexible agreements
- Thousands of hardware and software certifications
- Extended, long-term support
- Enterprise-grade systems for fixed function applications
- Ease of customization in order to deliver Just Enough Operating System [JeOS]¹⁴

SUSE Embedded Solutions enable developers to come to market faster and provide competitive advantages through customization. Read the Teradata case study at: [www.suse.com/success/stories/teradata.html](http://www.suse.com/success/stories/teradata.html)

Learn more about a SUSE Embedded Solution by visiting [www.suse.com/partners/embedded/](http://www.suse.com/partners/embedded/) or contacting our technical team at: [embedded@suse.com](mailto:embedded@suse.com)

---


---

**Exhibit 4. SUSE Linux Enterprise service security components**

- **Security Patches and Updates**
  over the whole product lifecycle

- **AppArmor**
  for fine-grained security tuning

- **Intrusion Detection**
  using AIDE

- **Linux Audit System**
  CAPP-compliant auditing system

- **Security Certifications**
  like FIPS, CCC/EAL4, etc.

- **SUSE Firewall2**
  Easy to administer OS firewall

- **OS Security Guide**
  Covering all security topics

- + more