Decomposing System Security Requirements

Trusted Computing: The COTS Perspective Series

Read About

Identifying what to protect

Recognizing system threats

Creating a system security plan

Introduction

A system's security requirements are determined based on a variety of factors, including (but not limited to) the program and application for which it's being developed, national regulations, and, sometimes, IP protection procedures. But once a system's essential security level is defined, what's next? How are top-level security requirements then translated into system- and subsystem-level Trusted Computing capabilities? How is a comprehensive security plan developed, implemented, and verified to meet requirements?

Managing system security requirements calls for multiple iterations and collaboration to identify risks and vulnerabilities in the system and develop mitigation strategies.

In our previous paper, The Impact of Protecting I/O Interfaces on System Performance, we considered the importance of a system's various interfaces and how ensuring security at these potentially vulnerable entry points may introduce performance trade-offs that must be mitigated, or at least recognized and addressed. In this white paper, we will discuss the process of decomposing high-level requirements for system security in order to develop an entire system security protection plan.

Following a Framework

A system's security requirements are often defined using criteria from multiple sources, such as the initial RFP or reference documents that provide system design guidance. The best approach for dealing with security requirements is to follow a framework that determines the process you will use based on the security levels that you are targeting. This framework will not tell you what you need to do to implement the required security, but will instead lead you through the thought process that you need to apply.

For example, the Committee on National Security Systems (CNSS) provides direction about which controls to consider in order to protect National Security Systems at a specific level of confidentiality, integrity, and availability. Similarly, the CNSS provides a path for developing an architecture for authorization capabilities or message integrity. Although some security guidance documents for DoD systems are classified, there are similar sorts of documents that instruct the system security designer about what types of things they need to think about as they go through the architecture and design process for their system.

Performing a Risk Analysis

After a framework, or set of frameworks, for realizing the system's high-level requirements has been established, an iterative, collaborative process between the customer and integrator is required to reach an agreement about how best to meet those requirements. This process considers both the context of the system and the use cases in which it will operate, and involves a risk analysis that looks at where and how the system will operate. For example, will there be armed guards always present or will the system be left unattended for weeks or years? The risk analysis evaluates how these use cases can affect the potential risk of all the different types of attacks that the security is meant to protect against. Normally, the risk analysis isn't quantitative, based on a formula that results in a risk number, but more of an engineering risk assessment that informs the designer where in the system architecture they need to apply necessary mitigations.

The first step in the risk analysis process is to create a high-level logical description of the system to model the data and flows in the system. Again, this should be an iterative process, and is optimally performed while the physical implementation of the system architecture is still in the process of being defined. That allows the security implementation of the architecture to mold the implementation, and vice versa, in order to obtain the optimal tradeoff.

Due to their likely influence on the physical implementation of the architecture, it's important and valuable to have the System Security Engineers (SSEs) active and involved at the right phase of the development cycle. Having them involved early on increases opportunities for beneficially influencing design trade-offs. After the physical implementation is set, there are greater limits on what security mitigations can be applied to the system, making any needed modifications more difficult and costly. It's also possible that these limits force the system designers to accept some vulnerability will be left in the system, to be dealt with in the next generation.

Identifying What to Protect…

The next step in the process is for the system security architect to determine what needs to be protected, such as specific algorithms, data, interfaces, and capabilities.

For example, sometimes you want to protect algorithms from being viewed or modified, especially if the algorithms need to be shared between systems or software developers. In some systems, it's the actual data that needs to be protected, such as in a system that is receiving signals intelligence that unauthorized users shouldn't be able to view.

For another example, consider UAV mission data that must be protected from remote hacking. There are different aspects of the system that need to be protected depending on what the system is designed to do, what the use cases are, and what sort of environment it will operate in. The security team also has to consider the classification levels of other systems with which the system will communicate. All of these issues and more come into play when deciding what needs to be protected.

…And What to Protect It from…

Once you've determined the elements of your system that require protection, it's important to determine what threats need to be defended against. For example, the failure of a safety-critical system can lead to catastrophic consequences, meaning the continuous availability of a safety-critical system must be guaranteed. For this reason, Denial of Service attacks that threaten to bring the entire system down are a significant danger and must be thwarted.

…And How Attacks Will Be Carried Out

After determining what aspects of the system warrant protection, the next step is to analyze the potential origins of the types of attacks which are expected and likely for that system. Are they remote attacks, local attacks, or insider threats? Will potential attackers be typical hackers or resource-rich organized criminals or nation states?

To protect against an insider threat, for example, it's important to ensure that the system software was developed without a backdoor route into it. As well, hardware must be free of nefarious components that are sub-optimal or fails if someone sends a key phrase. To offer this level of assurance, the entire hardware supply chain must be tested and verified.

Remote attacks, on the other hand, can include an attempt to access the video feed from an unmanned platform. If the system is connected to the internet and available, an attacker may have nearly unlimited time to try to gain access to the system data in order to modify its operation or deny availability. If an adversary gains physical access to the system, even if there are no external connections, efforts might be made to fool the system into believing that it's in an operational environment when it isn't. Local attacks can be as simple yet dangerous as a malicious actor plugging in a USB key to capture all the data.

Mapping Logical Attacks to Physical Components

In order to identify potential physical avenues of attack, the SSEs must move from theoretical analysis to considering what the system will actually look like. This involves taking the system logical diagram that defines what the system does and what functions need to be performed, and decomposing that information to an actual physical architecture that identifies on which board (or board sets) those functions will reside and how those modules will communicate with each other. From here, the backplane and network interfaces, including how data will flow throughout the system, can be defined. This process enables the SSEs to map the actual physical system operation to the potential attack vectors that were already identified, and possible remediation can be proposed for each one.

Defining a Final System Security Plan

It's at this point that the iterative process between the system developer and the customer begins. It's not uncommon for this process to lead to changes in the system's physical architecture. For example, it may be necessary to remove some connections from a single board computer to eliminate vulnerabilities, boost security by adding an encryptor into a data path, or add an encrypted hard drive to store data. Another option might be to apply some additional logical constraints that provide sufficient remediation, thereby changing the security aspects of the way the system operates in order to mitigate a threat instead of making changes to the physical architecture. For example, software can be implemented to periodically perform a certain activity that verifies the current functionality of the system.

Taking a 360° View

As your system architecture evolves to incorporate new security mechanisms, it's critical to assess if any new vulnerabilities have been created. In curing known issues, have any new avenues of attack been introduced and is there any new remediation that needs to be applied?

As well, you need to consider the trade-offs of implementing security protections, including the effect that any remediation might have on system performance. And, while identifying which techniques can be applied to protect against potential attacks, it's important to weigh whether the risk of each type of attack is worth the cost of remediation. The cumulative risk of different attack vectors actually being exploited versus the cost of applying the remediation technology into the system needs to be understood, since it influences how best to proceed. For example, a single cost-effective remediation technique might protect against a wide variety of attacks. On the other hand, if there is a potential attack vector that requires applying a very expensive technique, there might be less justification. Risk analysis might argue that a potential type of attack isn't very probable, or that it would only likely come from a group that would have very little chance of access to the system because of how it will be operated. Then SSEs might decide not to apply it, and instead go with a less costly method that isn't as effective. In some cases, depending on the risk analysis and related costs of remediation, it might be decided to leave a low-risk potential attack vector as an acceptable residual risk and well-understood vulnerability.

Additional remediation might also instead be applied to processes and procedures. For example, after being deployed, the system can be checked on a regular basis to ensure that it's still in good shape.

Considering Real World Context

In an ideal world, all of the groups involved in developing the system would work together through this iterative process. Unfortunately, that's not always possible. Realistically, different groups are available at different periods of time. Contracts may start and stop, which can cause discontinuity in the groups working on the system architecture. Additionally, security considerations might not be addressed until after the physical architecture has already been defined, meaning that security must be layered on top of existing architecture.

On the government side, there is an authority assigned to the system that must accept the design and all the risks that have been identified. Often, this authority is disjoint, with one government group responsible for cybersecurity, one for anti-tamper, and another for system certification, for example, depending on the specific requirements and from where they flowed down. For all of these reasons, it's crucial to get the SSEs involved as early as possible in the architectural discussions, before the architecture gets finalized. Similarly, the security team should start engaging their suppliers, such as commercial off-the-shelf (COTS) hardware vendors like Curtiss-Wright, as early as feasible. This is so the SSEs can both understand what solutions are available and engage in discussions about what steps can be taken to accomplish the security and performance goals of their system.

Not All Solutions Are Created Equal

When defense organizations, aerospace companies, and system integrators are evaluating embedded computing solutions, it is extremely important to understand exactly what vendors mean when they say their solutions provide Trusted Computing. It is even more important to understand the difference between solutions that provide Trusted Computing and those that offer a TrustedCOTS™ level of protection.

Curtiss-Wright TrustedCOTS solutions extend Trusted Computing best practices to every part of the development process, from design and testing to supply chain and manufacturing. The highest possible levels of protection are built into every aspect of solution development to increase the overall value that COTS solutions can provide in a secure system. This process includes careful analysis of the relationships, intersections, and dependencies among all of the various protection domains, and investigation into potential faults and failures at the lowest levels to identify the associated security vulnerabilities.

Curtiss-Wright takes a holistic view of Trusted Computing, going above and beyond the efforts of other vendors to apply the advanced protection capabilities needed to develop truly secure COTS solutions. It's one of the main reasons the company has been a trusted, proven leader in the global defense and aerospace industries for decades.

Conclusion

A system's security requirements are influenced by a variety of different factors, guidelines, and policies, and decomposing these system security requirements into an implementation plan is an iterative, collaborative process between customers and system integrators. A comprehensive approach involves not only establishing what data and components must be protected and what types of attacks must be protected against; it must also ensure that, when incorporating security mechanisms to defend against these established threats, no new vulnerabilities have been created.

What's more, like any other program requirements, security specifications must be balanced with and considered against other constraints, such as cost. Ultimately, defining a security plan requires a 360° view of your system's use case, capabilities, constraints, and trade-offs. Involving COTS hardware vendors, like Curtiss-Wright, early on in the process is helpful in determining what solutions are available and how exactly security goals can be achieved.

Author(s)

David Sheets
Senior Principal Security Architect
Curtiss-Wright Defense Solutions

Learn More

Curtiss-Wright Products

- Data at Rest Protection
- Security Enabled Curtiss-Wright SBC and DSP Products

Curtiss-Wright Case Study

- Building a Truly Trusted Computing Solution with COTS Hardware and Intel Security Capabilities

Curtiss-Wright White Papers

- The Many Faces of Trusted Computing: What You Need to Know to Protect Critical Platforms and Data
- Getting Secure, Intel-Based Solutions to Market Faster - Why the Hardware Vendor's Boot Security Implementation Is So Important
- Beyond Trusted Computing: Extending Protection Capabilities to Deliver TrustedCOTS Solutions
- Trusted Computing: The COTS Perspective Series
  1. Introduction to COTS-based Trusted Computing
  2. Trusted Boot
  3. Hardware Features for Maintaining Security During Operation
  4. Considering the Role of Hardware in Securing OS and Hypervisor Operation
  5. Application Development, Testing, and Analysis for Optimal Security
  6. Developing a Secure COTS-Based Trusted Computing System
  7. The Impact of Protecting I/O Interfaces on System Performance
  8. Decomposing System Security Requirements
  9. Establishing a Trusted Supply Chain
  10. Certification Authorities for Trusted Computing in Military and Avionics Products