White Paper **WP090001EN** Effective May 2021

Eaton cybersecurity design principles: Recommendations and best practices

Ashim Dutta, Lead Cybersecurity Engineer, Eaton

Prateek Singh, Lead Cybersecurity Engineer, Eaton

In this white paper, we present Eaton's cybersecurity design principles, which are based on multiple industry standards and best practices, including IEC62443, UL, and OWASP. We take cybersecurity seriously. Collaboration with standards-making organizations and adherence to global standards will in the long run ensure that only trusted software or firmware is deployed, a must in a world that becomes more connected and electrified every day.

As more manufacturers and industries build and deploy smart industrial internet of things (IIoT) devices, the security of systems become more important and difficult to manage. We know that trusted equipment is the backbone of safe network environments. Accordingly, we have adopted a "secure-by-design" philosophy for our products.

Our cybersecurity design principles form the foundation for secure development of all intelligent products at Eaton, and our secure software development life cycle (SDLC) process integrates security at every phase of the product, from its inception to deployment and maintenance phases. These 13 design principles have been selected based on industry standards and best practices.

Learn how Eaton is taking the lead in cybersecurity at: eaton.com/cybersecurity

Powering Business Worldwide

Minimize attack surface area

The attack surface—the areas within a product or system that could provide opportunities to exploit in a cyberattack—grows larger with every addition of new features, such as adding new network protocols or webpages. Whenever a new feature is added to an existing product or a system, it increases a certain amount of security risk to the overall system. The attack surface area of a product or a system will always increase with time as an organization adds more interfaces or features to the product or system to integrate with other products or systems. However, from a security perspective, it is very important that an organization looks for ways to reduce and minimize the overall size of the attack surface by various means possible.

The aim of cyber secure development is to take proactive steps to reduce the overall attack surface area of the product or system when changes are made. In a product or system, this can be done with steps such as:

- Reducing the number of user roles
- Avoiding storage of confidential data that is not needed
- Disabling features that are not needed or not needed all the time
- Introducing operational controls such as a Web Application Firewall (WAF) and other intrusion detection and prevention systems

For example, a product might use various open ports and services for operation. However not all ports and services may be needed or even used at all times. You can reduce the attack surface by removing unwanted ports and services from the product, making it more secure.

Similarly, it is important to remove debug ports, headers and traces from circuit boards used during development from production hardware. Such additional mechanisms may be exploited by malicious users to compromise the product or system by dumping the firmware or extracting critical information such as keys and passwords.

Secure by default

End users tend to prefer out-of-the-box convenience, but such user-friendly convenience can introduce security risks. For organizations that deliver out-of-the-box solutions, aiming for "secure by default" is critical. A product or a system is considered secure by default when the default factory settings for the system are secure. For example – all ports and services not needed for the running of the product should be turned off by default. This means that organizations and its product developers must attempt to make the product/system usable with all security mechanisms in place. This "usable security" principle ensures that security features aren't circumvented for the sake of convenience and recognizes that usability and human factors are key components of secure design.

For Eaton products, "secure by default" means shipping the product in the most secure (yet usable) state at the outset, requiring minor user configuration to achieve a reasonable level of security. Configuration options to achieve this goal may include:

- HTTPS-only by default
- Immediate password change prompt on first login
- Disabling all insecure services

Some cybersecurity standards and regulations (such as UL2900 and IEC62443-4-1) may also require some security features enabled by default, such as password aging or password complexity.

Of course, end users may choose to modify the security settings (if allowed) or even disable them. The end users must be made aware of the risks of taking such actions. This can be as simple as a pop-up message should a user modify a security setting as documented in the product's secure configuration guidelines.

Least privilege

The principle of least privilege is one of the oldest pillars of secure design. It recommends that within any product or system, the services, processes or users should have the least amount of privilege required to perform their operations normally. This includes such things as:

- User rights
- Resource utilization
- Resource permissions (*e.g.*, CPU limits, memory, network and file system permissions)

Every process in the product/system must operate with the least privileges needed for it, both in terms of resources that can be accessed by the process as well as the duration of time for which those resources can be accessed by the process. For example, in a Linux system, all processes should not run with root privileges by default. Similarly, in Microsoft Windows, all applications should not always need Admin system privileges to run.

The principle also recommends that mechanisms used to control access to resources must never be shared. For example, if an application on a server requires only access to the network, read-only access to the database and a read-write access to the audit log file, then the server should be granted only these permissions and nothing else. The application should never be granted administrative/system privileges.

When it comes to user roles in the product/system, least privilege is a responsibility of both the manufacturer and the operator. While it is the responsibility of the manufacturer to design products and systems with this principle in mind, it is also the responsibility of the operators to configure users and roles within the product while keeping this principle in mind.

Defense in depth

The principle of defense in depth suggests that instead of a single security control (that may still be reasonable security control), it is always better to have more security controls deployed in a layered manner.

A defense-in-depth design strategy involves designing and implementing multiple layers of security controls around a product or a system in such a way that an attacker would have to compromise multiple levels of security in order to gain access to the most critical resources. This layered approach to resource authorization requires unauthorized users to circumvent each authorization attempt to gain access to a resource. This can often discourage a cyber-attack on the product as it would significantly increase the time and effort required by an attacker to compromise the system.

For example, an industrial automation and control systems (IACS) product is much less likely to be compromised if it not only has access control on it, but is also:

- Installed on a separate network
- Firewalls are placed between different networks
- Security events and accesses are logged at the product, system and network level

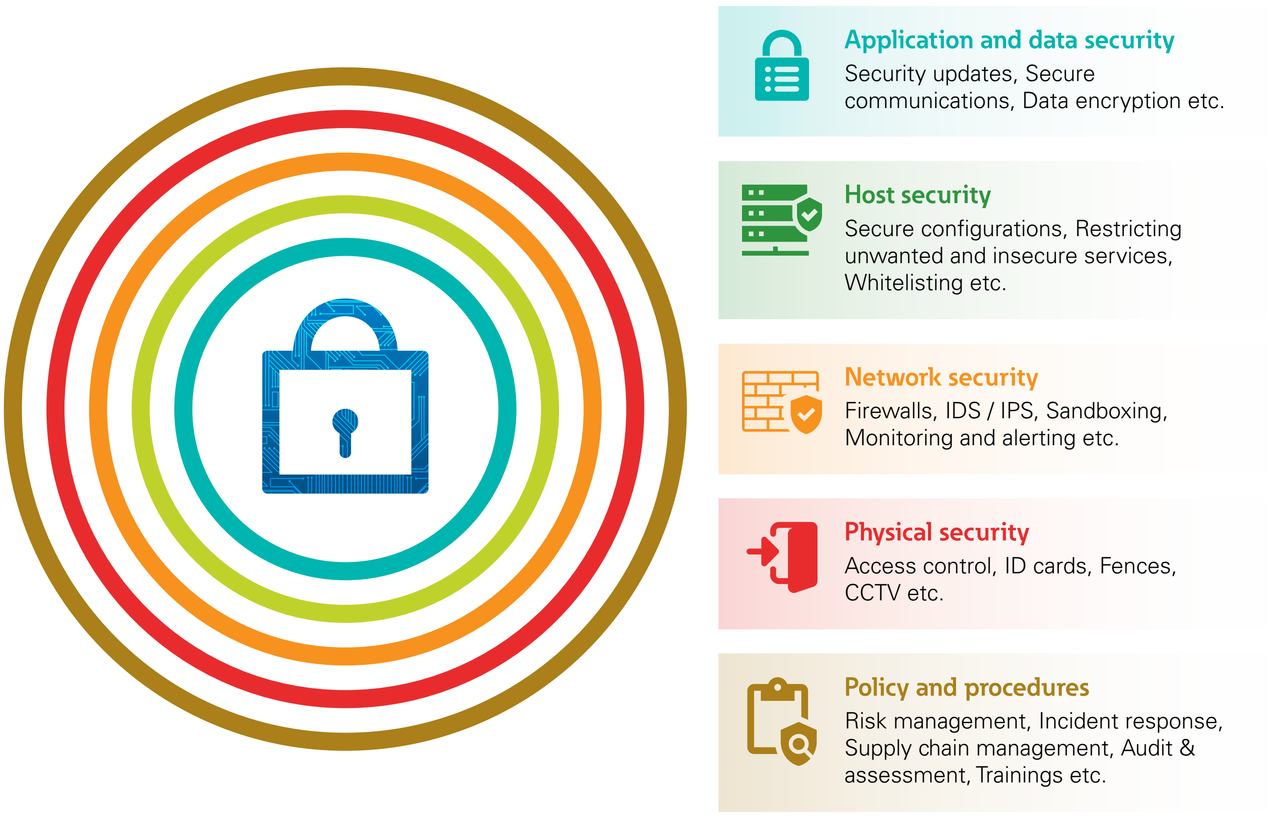
Application and data security
Security updates, Secure communications, Data encryption etc.
Host security
Secure configurations, Restricting unwanted and insecure services, Whitelisting etc.
Network security
Firewalls, IDS / IPS, Sandboxing, Monitoring and alerting etc.
Physical security
Access control, ID cards, Fences, CCTV etc.
Policy and procedures
Risk management, Incident response, Supply chain management, Audit & assessment, Trainings etc.

Figure 1. Defense in depth layers

Defense in depth is the responsibility of both the manufacturer and the customer. As before, while it is the responsibility of the manufacturer to design products and systems with this principle in mind, it is also the responsibility of the customer to configure the system in the field with this principle in mind.

Such security controls when used in a layered manner with different depths, can often make severe security vulnerabilities difficult to exploit in a real system and hence minimize the overall damage as well as make a full compromise much more unlikely to occur.

Fail securely

At some point, a product, system or application may fail to correctly process a function or request. The fail securely principle means that when such a failure occurs, and for whatever reason, the failure occurs in a secure manner. In other words, there should be a defined failed state which should not compromise the security of the system. For example, system failure should not automatically:

- Provide additional user privileges
- Permit the user to bypass access controls
- Allow access to sensitive user information
- Allow access to keys and passwords
- Allow attackers to install unauthorized software
- Allow attackers to use the product to launch further attacks on the customer network

Avoid security by obscurity

The principle of security by obscurity relies on concealing or limiting exposure to access credentials, such as encryption keys, certificates or passwords. Keeping this information secure is important, of course, but relying solely on restricting this information to a few users is likely to fail at some point, especially if security by obscurity is the only control measure.

For example, the security of a product should not rely on an SSH port that can be enabled by users accessing a certain command or an API, which can then be used to access and even modify all stored credentials. The security of an application should not depend on the source code of the application as a secret, which can be easily reversed in some cases to expose valuable information about the application, including keys. Instead the product or application security should rely on multiple factors, such as passwords and password policies, defense in depth, and audit controls.

Some embedded products may have dip switches that allow users to bypass the default security of the products in certain emergency cases. The customer-facing product documentation may not include this information because the manufacturer is relying on security by obscurity, which will work only as long as the bypass mechanism is not publicly known.

Another example is Linux and its open source code. Because the code is publicly available, security by obscurity does not work here at all. If Linux is incorporated into products, it is important to deploy additional controls so that the product can be more robust and cyber secure.

Keep security simple

Sometimes product and system manufacturers choose overly complex methods, procedures or system architectures to implement security rather than choosing straightforward methods. The more complex the architecture, the greater chance that errors can occur. The simpler approach may require a bit more effort due to possible re-design of the product, but it will be more secure, providing advantages in the long run.

Simple security and minimizing attack surface area principles go hand in hand. The focus should be on protecting information rather than simply relying on policies and procedures. Implement access controls at every point to protect against insider as well as outsider threats. Security needs to be integrated into the design process and it cannot be an afterthought. Do not add functionalities that increase the security risk of the product unless they offer significant value for the end-user. Do not add functionalities that can be met by other methods. For example, do not allow remote access (SSH) with full access by default just to access the internal audit logs. Instead, provide an option to the user to directly access the audit logs from an authenticated product interface without having to log in to the device using SSH. Manufacturers of multiple products should have a single common and secure configuration tool instead of having a tool for each product, working over different interfaces and developed without security in mind.

Use secure components and designs

For product and system development, only proven designs or design patterns should be used. Developers should use industry standard and approved cryptographic algorithms (such as the National Institute of Standards and Technology (NIST) and the Federal Information Processing Standards Publication (FIPS)) and modes of operations for security protocols must be used for all features, such as encryption or firmware authenticity checks. As long as a proven and secure design solution exists, developers should avoid creating their own algorithms.

When using third-party open source code or commercial off-the-shelf (COTS) hardware or software, it is important to ensure that only secure components are used wherever possible. Validate that firmware and application software contain code only for required functions—identify and remove dead code. And be sure to remove any debug or development features (hardware or software) from products or systems in production.

Security documentation

Documentation is a critical and must not be ignored when it comes to security. There should be adequate documentation to cover and support the entire life cycle of the product. All documentation should be prepared by the team responsible for designing and developing the product. Documentation can be categorized in three types: Internal, maintenance and end use.

Internal documentation should be prepared with the goal to provide full understanding of the product, which in turn helps OEMs in debugging, adding feature enhancements or fixing any identified cybersecurity vulnerabilities. This documentation remains with the OEMs. Internal documentation should include, but is not limited to:

- All functionalities
- Technologies used
- High- and low-level designs
- Software and hardware architectures

Maintenance-related documents should contain details to help and support supervisors, field or maintenance engineers to deploy or commission a product in a secure way. Maintenance documentation should include, but is not limited to:

- Installation processes
- Secure commissioning and deployment
- Secure configuration
- Troubleshooting guidance
- Update/upgrade processes

End use documentation should focus on secure usage and decommissioning of the product. This documentation should guide end users to use the product in a secure environment and to dispose of the product securely at the end of its life cycle. Documentation for end users should be accessible and provided with the product or published on the OEM's website. End user documentation should include, but is not limited to:

- Security hardening guidance
- Firewall rules
- Secure decommissioning procedures

Promote privacy

Now more than ever, privacy is a critical parameter that must be taken into account when designing products and systems. All customer data processed by a product or solution should be defined, including:

- Audio recordings
- Video recordings
- PII, data (personally identifiable information such as name, email, address, phone, etc.)
- SPII data (sensitive personally identifiable information, such as citizenship or immigration status, medical information, ethnic or religious affiliation, account passwords, etc.)

With the growth of IoT technologies, many products and systems provide advanced features to process data in cloud environments, and technical and customer data may be collected for the purpose of data analytics. However, the users of a product or a solution may not be aware that their private information is being collected and processed. Users must be given control of their private data with clearly written disclosures and clear opportunities to provide affirmative consent. This "opt-in" approach is very different than deploying a "default-on" setting that requires the user to take steps in order to opt-out of providing data.

For IoT applications (*e.g.*, intelligent motor management or cloud-connected PLCs, etc.), protecting user data privacy may mean implementing security mechanisms, such as: end-to-end communication encryption; encrypted databases; secure inter-cloud communications; and encrypted disks or virtual machines (VMs) as well as operational measures such as a request portal for secure deletion of data and documented policies on storage and deletion of customer data.

For embedded products (*e.g.*, intelligent electronic devices, automation controllers and electronic relays), this may mean the implementation of security mechanisms like encrypted disks, encrypted databases, or encrypted communications along with tamper-proof hardware and end user documentation about the secure decommissioning and disposal of the product (also known as "zeroization").

Organizations should never allow their products or systems to compromise the privacy of the user. Proper anonymization measures should be implemented to fully remove all user data or credentials stored on the product when needed.

Zero-trust security

"Zero trust" is defined by the U.S. National Institute of Standards and Technology (NIST) as a cybersecurity paradigm based on a "trust no one" approach. Trust is never granted implicitly to anything or anyone and must be continuously evaluated with an overall focus on resource protection.

Zero-trust security is a combination of connected and inter-linked policies, practices, software and hardware that together create a zero-trust ecosystem. What this means is that within a network, no user or product or application is trusted by default from either inside or outside of the network. Authentication and authorization are needed by anything or anyone trying to gain access to resources on the network.

Access into a system or a network—everything from inside or outside a network, to a product, or to a remote, on-cloud resource—should be authenticated and authorized to prevent access by unauthorized and unintended users. Proper access control mechanisms must be implemented in products at each layer (defense in depth) to prevent unauthorized access. Accesses and trust should be based on user role profiles, and it is important to distinguish between different types of users (*e.g.*, administrators, users or operators).

For cloud applications, two-factor or multi-factor (MFA) authentication, such as one-time password, tokens or biometric recognition, may be implemented. MFA requires more than one method to authenticate an authorized user. With layers of authentication methods independent of each other, this adds additional layers of defense and makes it difficult for malicious users to gain unauthorized access. For local network connections from external or public systems, a multi-factor authenticated virtual private network (VPN) connections may be implemented to provide secure communication. Local physical network connections may implement two-factor authentication using proximity devices (such as Bluetooth or other near field communication (NFC) devices) or physical buttons in addition to passwords.

Under no circumstances, should a system or a product allow direct unauthorized execution of externally provided commands, scripts or other parameters that are not within the defined functional scope of the system or product.

Input validation and output encoding should be performed on all user and machine input because input fields can be used to inject crafted malicious commands to disclose, corrupt or delete confidential/ sensitive information.

While we must take care of implementing zero trust concepts in Eaton products and solutions, any zero trust steps we take must also be supported by our customers when they deploy our products and solutions. We provide secure configuration and maintenance guidance with all our products and solutions so that our customers can deploy and operate our products in line with zero-trust security principles.

Software integrity

No system can be 100% safe. Even when products and systems are designed in adherence to cybersecurity design principles, there is the potential that new bugs or vulnerabilities are discovered after deployment. Therefore, it is important that organizations allow updates to the product software or firmware to ensure that the product is always patched to protect against new vulnerabilities.

However, if the update mechanism itself is not carefully and securely implemented, this itself can lead to additional vulnerabilities. In the absence of these protocols, malicious users may be able to install their own software or firmware in the product and disrupt its normal operation.

To prevent such scenarios, software or firmware updates for products should use standard cryptographic mechanisms to establish authenticity and integrity. This is generally implemented using digital signatures, such as asymmetric cryptographic techniques and public key infrastructure (PKI), which can be used to detect any changes to the integrity of the software or firmware against the original, pre-installed version.

Digital signature algorithms, such as RSA or DSA, are public key encryptions. In public key cryptography (asymmetric encryption), the authentication occurs using one private key that is paired with a public key. The older private key cryptography (known as symmetric encryption) a single key is used to encrypt and decrypt data transmission. In the case of IoT systems, you can see why private key cryptography introduces risk. If the key is not unique for all the products, which is often the case in IoT systems, then access to the private key may allow a malicious user to create valid firmware images for all such products in the field. This weakness is not present in public key cryptography.

It is also important to implement methods or controls to prevent a malicious user from installing older, vulnerable versions of the software or firmware. Hence integrity verification mechanisms must be incorporated into products for all firmware images, scripts, executables, critical configuration files and other important files included in the product. Some examples of integrity verification mechanisms include validating against hash values or verifying against digital signatures. It is recommended to verify the integrity at the start-up and at periodic intervals when a program executes.

Life cycle support and vulnerability management

With the rapid growth of IoT and IIoT, product life cycle support is a critical area that organizations need to address. Malicious users actively look for areas of weakness to launch cyberattacks, and the need for vigilance never ends

Organizations should implement a vulnerability management program that regularly monitors and addresses security vulnerabilities in its product throughout its life cycle: prior to release; during release; and throughout the supported lifetime of the product in the field. Organizations may choose to deploy continuous vulnerability monitoring tools across the hardware and software application portfolio for this purpose.

Organizations should define how to apply security updates to the product, either over the air or locally, wherever possible. Organizations also need to ensure that such updates are checked for authenticity and integrity using the United States' Federal Information Processing Standards Publication (FIPS) approved cryptographic methods before installation and deployment in the field. And organizations should take steps to implement "anti-rollback" safeguards to prevent the installation of older, vulnerable versions of firmware or software.

Organizations should support IoT products through the four stages of the life cycle: provisioning, configuration, maintenance and decommissioning. However, they should also be capable of verifying the identity of new and old components at any point of time and de-provision any component that does not meet the organizational policies or are at the end of IoT product lifecycle. This is applicable to products as well as applications in the system.

For IoT products with cloud-based solutions, service providers must have business continuity planning (BCP) which includes disaster recovery (DR) plan to support operational resiliency. A DR plan should have a clearly defined recovery time objective (RTO) and recovery point objective (RPO) to recover the product or solution with minimum damage sustained. Cloud-based solution service providers should also have adequate measures in place to meet the defined RTOs and RPOs.

References

Security by Design Principles by OWASP

https://wiki.owasp.org/index.php/Security_by_Design_Principles

IOT Security Top 20 Design Principles by UL

https://ims.ul.com/sites/g/files/qbfpbp196/files/2018-05/iot-security-top-20-design-principles.pdf

Secure Coding Principles by OWASP

https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf

NIST Special Publication (SP) 800-175B Revision 1, Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175Br1-draft.pdf

Eaton
1000 Eaton Boulevard
Cleveland, OH 44122
United States
Eaton.com

Powering Business Worldwide