Approaches for Embedded System Information Security (2010 Revised Edition)

Know Your Organization's Security Level by Checking 16 Points

September 2010

IT Security Center,
INFORMATION-TECHNOLOGY PROMOTION AGENCY, Japan

This page intentionally left blank.

Contents

Background of the Guide 1
Background of Revision 2
1. Preface 3
1.1. Current Status and Challenges Surrounding Embedded System Security 3
1.2. Purpose of This Guide 3
2. Terminologies 4
3. Security Level 5
3.1. Embedded Systems and Security 5
3.1.1. Internal Architecture of Embedded Systems 5
3.1.2. Attacks against Embedded Systems and Countermeasures 6
3.2. Lifecycle of Embedded Systems 7
3.2.1. Definition of Lifecycle 7
3.3. Security Levels 9
3.3.1. Management Policy 10
3.3.2. Planning and Development Policy 11
3.3.3. Operation Policy 12
3.3.4. Disposal Policy 13
4. Security Measures at Each Phase 14
4.1. Management 14
4.1.1. Security Rules 14
4.1.2. Security Education 16
4.1.3. Collecting Security Information 18
4.2. Planning Phase 19
4.2.1. Budgeting 19
4.2.2. Selecting Development Platform 20
4.3. Development Phase 22
4.3.1. Designing 22
4.3.2. Software Implementation 28
4.3.3. Outsourcing 30
4.3.4. Security Assessment Test and Debugging 32
4.3.5. User Guide 34
4.3.6. Factory Production Control 37
4.3.7. Support of New Technology 39
4.4. Operation Phase 42
4.4.1. Handling Security Issues 42
4.4.2. User Notification and Fixing Vulnerability 43
4.4.3. Leveraging Vulnerability Information 45
4.5. Disposal Phase 46

4.5.1. Promoting User Preparation for Disposal 46
Appendix 49

Figures

Figure 3-1 Internal Architecture Model for Embedded System. 5
Figure 3-2 Examples of security problems at each phase of the lifecycle 7

Tables

Table 3-1 Threats against Embedded Systems and Countermeasures 6
Table 3-2 Examples of Information Assets Embedded System Should Protect 8
Table 3-3 Security Levels at Each Phase of Lifecycle. 9
Table 4-1 Examples of Typical Side Channel Attacks 25

Background of the Guide

In these years, a network environment where you can bring various types of embedded systems into it has become a growing reality. It is expected that embedded systems will keep evolving with higher functionality and more multifunctionality, and become literally embedded into every aspect of the society. While this happens, a major concern has also been addressed that the adoption of off-the-shelf, general-purpose software used for PCs and servers into developing embedded systems for cost and time reduction and increased productivity may harm the level of the quality and safety required for embedded systems. Moreover, because of their networked environment, embedded systems now must face the same cybersecurity threats as PCs do as well. For malicious cyber attackers, embedded systems that are closely entwined with money-related information can be a perfect target. For those who wish to cause chaos in the society, the embedded systems are a desirable target as one of the social infrastructures.

It is becoming more and more important that establishing and implementing a strategic approach that enables and encourage embedded system developers to implement necessary security measures based on what we have learned with security incidents over PCs and servers.

In reality, there is no perfect security and the balance between security and cost usually comes into play. To find the adequate balance between them, the developers must properly know the current security effort and level of their organization. Especially for the embedded system projects, it often happens that the time schedule is severely tight and the required specifications and constraint conditions change during the development. To overcome these difficulties, a concrete, specific security guide that helps the developers sort out security problems and improve security of their product.

By using this guide as a reference, the developers will be able to know the security level of their organization and see what need to be done to achieve a higher level.

Here, IPA provides a guide that serves the above purpose as an output of the Project on Information Security Technology for Embedded Systems funded by Japan Science and Technology Agency.

In addition to ensuring reliability and safety, we must also discuss and adopt security measures that protect embedded systems from malicious attacks.

IT Security Center, Information-technology Promotion Agency, Japan
Security Engineering Laboratory Director, Hideaki Kobayashi

Information-technology Promotion Agency, Japan	(Ubiteq, Inc. *)	Senei Akabane
Information-technology Promotion Agency, Japan	(Hitachi, Ltd. *)	Masaharu Ukeda
Information-technology Promotion Agency, Japan	(Hitachi, Ltd. *)	Makoto Kayashima
Information-technology Promotion Agency, Japan	(IT Doraku Research Laboratory, Ltd. *)	Motohide Soeno
Information-technology Promotion Agency, Japan	(Toppan Printing Co., Ltd. *)	Satoru Takahashi
Information-technology Promotion Agency, Japan		Manabu Nakano

*As of June 2009

Background of Revision

A market for embedded systems, especially for information home appliances like digital TV with the Internet connectivity, is rapidly growing. Information home appliances are under severe development pressure and it is becoming common that the developers adopt the Web browser functions that have high usability and is easy to develop, and off-the-shelf, general-purpose technology like the Web server functions as an infrastructure to create a configure interface.

Meanwhile, the emergence of new network environments, such as IPv6 (Internet Protocol Version 6) and NGN (Next Generation Network), or the countdown to transition to the next-generation encryption algorithms gives embedded system developers heads ups on yet another issues and technologies they need to consider and adapt to.

Moreover, the information home appliance market is global and it is expected that the products will be sold worldwide. This means that the information home appliances may become a potential target of cyber attacks from all over the world, and once a security incident breaks out, the product developer has the responsibility of responding to the incident for the worldwide customers.

IPA revised the guide incorporating the feedback on the first issue, and also added technologies needed in developing information home appliances for worldwide use, especially to safely use advanced technologies, such as IPv6 and Web technologies.

IT Security Center, Information-technology Promotion Agency, Japan
Security Engineering Laboratory Director, Hideaki Kobayashi

Information-technology Promotion Agency, Japan	(Fourteenforty Research Institute, Inc.)	Yuji Ukai
Information-technology Promotion Agency, Japan	(Hitachi, Ltd.)	Makoto Kayashima
Information-technology Promotion Agency, Japan	(Hitachi, Ltd.)	Chisato Konno
Information-technology Promotion Agency, Japan		Manabu Nakano
Information-technology Promotion Agency, Japan		Tomoka Hasegawa

Writing Partners

OKI Networks Co., Ltd.	Yutaka Ibuki	Nippon Telegraph and Telephone Co.	Tomohiro Fujisaki
Nippon Telegraph and Telephone Co.	Kazuhiko Ohkubo	BUFFALO Inc.	Tatsuo Inagaki
Nippon Telegraph and Telephone Co.	Hiroki Tanaka	Panasonic Corporation	Kazutaka Maeda
Nippon Telegraph and Telephone Co.	Takemi Nisase	Panasonic Corporation	Masao Ito

Interview Collaborators (Individuals)

SHARP Co.	Takuya Ohkubo	Toshiba Co.	Satoshi Ozaki
Sony Co.	Tadashi Morita	Hitachi Consumer Electronics Co., Ltd.	Yoshihiro Yamada
Sony Co.	Masakazu Kobayashi	Hitachi, Ltd.	Satoshi Sano
Toshiba Co.	Tadahiro Aihara		

Interview Collaborators (Organizations) Mitsubishi Electric Co.

(In no particular order, titles omitted) As of September 2010

1. Preface

1.1. Current Status and Challenges Surrounding Embedded System Security

As semiconductors, such as microprocessor and memory, improve in performance, industrial equipments and information home appliances are becoming more and more high-tech. Embedded systems literally embedded to control those devices have been used in a wide area. As information communication technology advances, embedded systems also become connected to open networks like the Internet. This exposed the embedded systems to a new risk: cyber attacks by malicious attackers just like PCs.

If embedded systems are connected to the Internet, they also have to adapt to new technologies the Internet is built and operated upon. For example, to respond to IPv4 address depletion, the global transition to IPv6 is currently underway. Given this situation, the developers are required to support both IPv4 and IPv6 to develop embedded systems that communicate using the Internet or those that perform remote update of firmware, and monitoring and operation.

As the variety of services keeps expanding, such as high-vision broadcasting, online shopping and private information management using the high-tech weight scale, the value of information embedded systems work with becomes higher. For this reason, implementation of security measures against cyber attacks, such as preventin of unauthorized data duplication and attacks via network, and privacy countermeasures like deleting privacy information when disposing the product are becoming required.

An extremely short development period that embedded system developers are forced to work with under severe development pressure, however, tends to push the balance between security and cost toward the latter. The developers, not just programmers, but also managers and management executives, need to tackle the issue and find how to keep the good balance.

1.2. Purpose of This Guide

This guide provides the embedded system developers how to tackle the issue - what to do and the security level for each effort - to enable the development of secure embedded systems in the following aspects: (1) management of the development team, (2) planning and development of a embedded system, (3) operational requirements that the developers must take care of when the embedded system is in use, (4) requirements when disposing the embedded system. The guide targets the following people that work for the companies that develop embedded systems.

- System developers
- Project managers
- Decision makers for budget and resource allocation (management)

This guide aims to:

(1) Improve security awareness of embedded system developers

Helping the developers understand security required for embedded systems, realize the security level of the current security measures adopted, and learn what to do to improve the security level.

(2) Encourage implementation of better security into embedded systems

Helping the developers assess the security level and implement more advanced security measures to improve security of their embedded systems. Depending on the functionality their product offers, the developers can set the target level at 3 or 4. By practicing secure implementation, they can not only protect their users from falling victim but also protect the others that may suffer damage if the embedded systems are exploited to attack others. This also helps to avoid the cost of responding to the security incidents and protect the corporate brand from harm.

2. Terminologies

- Embedded System

  A specialized computer system to control the products. It consists of semiconductors and peripheral devices and is embedded into another product, such as industrial equipments and information home appliances.

- Product

  Those controlled by embedded systems, such as industrial equipments and information home appliances.

- Vulnerability

  Security flaws in embedded systems that could be exploited by unauthorized access or computer viruses and allow the degradation in functionality and capability of the product.

- Security Rule

  Security policy that concretely defines what must be done and what must not be done to make the organization’s security effort systematic instead of ad-hoc.

- Anti-Tamper

  Tolerability against attacks that try to access the module’s internal data without proper authorization by accessing it through the way other than the interface prepared by the module.

- Privilege Escalation

  Operation of changing the access control level of a system user to a higher one on the systems that have multiple settings, such as user privilege and administrator privilege.

3. Security Level

3.1. Embedded Systems and Security

3.1.1. Internal Architecture of Embedded Systems

Embedded systems vary depending on the product it is to control, from a small system with one-chip microcomputer to a large system with multiple microprocessors. In this guide, the internal architecture and external interfaces of the embedded system are modeled as below (Figure 3-1) to analyze security measures.

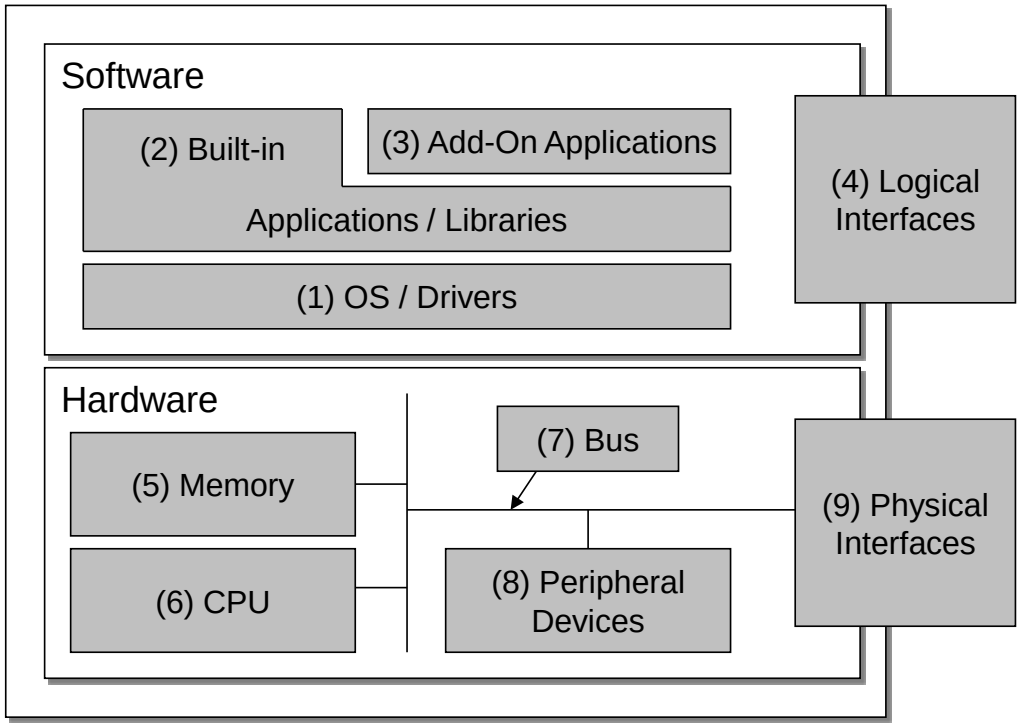

Figure 3-1 Internal Architecture Model for Embedded System

An embedded system consists of software, which in turn consists of some OS and drivers ((1) in Figure 1), built-in applications and libraries running on them (2), and arbitrary, add-on applications the user can install (3), and hardware in which software is installed. Hardware consists of memory units (5) and CPU (6), peripheral devices like TPM (Trusted Platform Module) (8), and buses to connect them (7).

In addition, an embedded system is connected to external systems, such as other systems and memory units like a USB memory stick and SD card, through various connectors, ports, physical interfaces like a touch panel display (9) and logical interfaces for various protocols (4).

3.1.2. Attacks against Embedded Systems and Countermeasures

Because of their ever-growing connection to the Internet during recent years, embedded systems have become threatened by the same cybersecurity issues as PCs are. For the safe and secure use of the embedded systems, the developers are required to implement countermeasures that assure at least the following three pillars of security.

- Confidentiality: to prevent the disclosure of the confidential information to unauthorized individuals
- Integrity: to prevent modification of data without being detected while data are being processed, transmitted or in storage.
- Availability: to assure that the system serves its purpose and the information is available to the authorized individuals when needed.

Threats against embedded systems could be the attacks through physical or logical interfaces, or those through the reverse engineering by an attacker who obtained the embedded systems. To implement the three pillars of security above, the developers need to analyze potential threats the component modules described in 3.1.1 using some systematic methods, such as the STRIDE Threat Model[1], and consider countermeasures. Table 3-1 shows the result of the analysis on the potential threats against the embedded system's component modules and the countermeasures against item.

Table 3-1 Threats against Embedded Systems and Countermeasures

		Confidentiality		Integrity		Availability	
		Threats	**Countermeasures**	**Threats**	**Countermeasures**	**Threats**	**Countermeasures**
Software	**Logical Interfaces**	・ Spoofing during transmission ・ Data interception during transmission	・ User/server authentication ・ Communication encryption	・ Data modification during transmission	・ Data signature	・ Denial of service attack	・ Filtering
	Add-on Applications	・ Privilege escalation ・ Installation of malicious programs	・ Vulnerability countermeasures ・ Certificate verification	・ Software modification and unauthorized use	・ Vulnerability countermeasures ・ Logging	・ Denial of service attack	・ Filtering ・ Vulnerability countermeasures ・ Logging
	Built-in Applications/ Libraries	・ Privilege escalation ・ Analysis of software configuration	・ Vulnerability countermeasures ・ Software hardening				
	OS / Drivers	・ Privilege escalation	・ Vulnerability countermeasures	・ Privilege escalation	・ Vulnerability countermeasures ・ Logging	・ Denial of service attack	・ Vulnerability countermeasures ・ Logging
Hardware	**Memory**	・ Probing ・ Side channel attack	・ Data encryption ・ Anti-tamper	・ Data modification ・ Data destruction	・ Anti-tamper	・ Denial of service attack due to loss of integrity	・ Anti-tamper
	CPU						
	Buses						
	Peripheral Devices					・ Denial of service attack due to monopolization of bandwidth and/or processing power	・ Filtering, logging, prevention of resource monopolization
	Physical Interfaces						

In addition, when a product with vulnerability is sold and used, it is possible that an attacker who discovered it may blackmail its developer for money in return for keeping quiet about the vulnerability. To protect the users and themselves, it is critical that the developers must make sure that the system is secure before shipment and be ready to respond to a new vulnerability which may be found after shipment to keep the product safe and secure.

3.2. Lifecycle of Embedded Systems

3.2.1. Definition of Lifecycle

As mentioned in 3.1, it is required to implement the security measures, which are appropriate to the value of information assets the components of the embedded system handle when they process and transmit data, into each component. In doing so, it is effective to apply the analysis through the lifecycle of embedded systems[2,3]. This guide divides the lifecycle into four phases - planning, development, operation and disposal -, and describes points that the developers need to take care of throughout the lifecycle.

For example, in the case of an information home appliance, it is planned and developed by the product manufacturer, sold to the users through the retailer, used by the user and disposed by the recycler. At the operational phase, the user first sets it up and then starts using it after obtaining it. During the period the user uses the appliance, the user may update the software applications, it may be broken and fixed by the manufacturer or retailer, or it may be given or sold to another individual and set up again. The threats against embedded systems exist not only in the operational phase but in every other phase as well. The developers must solve all of them (Figure 3-2).

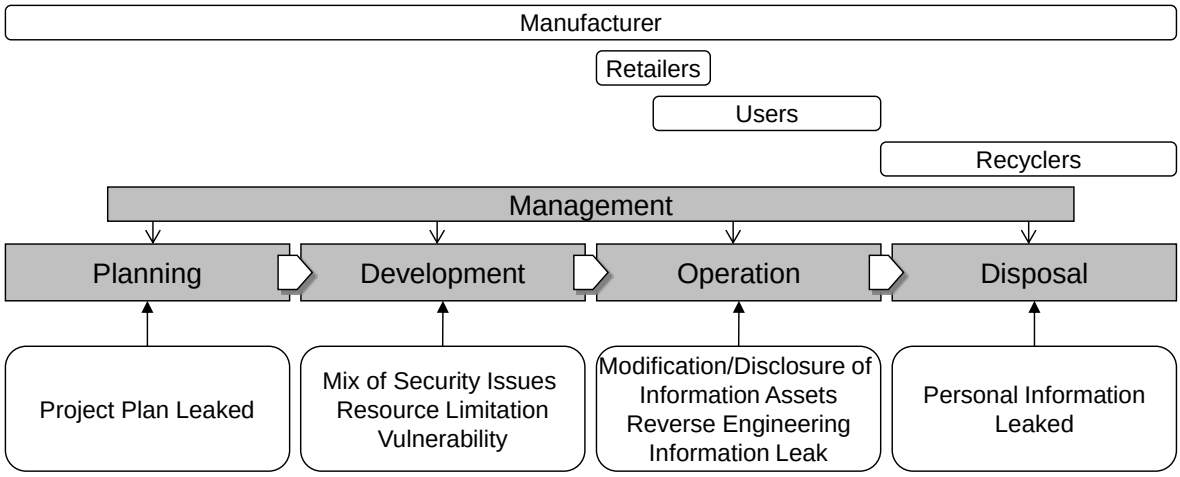

Figure 3-2 Examples of Security Problems at Each Phase of the Lifecycle

First, let's see the role and security issues of each phase of the lifecycle.

(1) Planning Phase

The planning phase is where an embedded system is planned and budget is discussed. It is this phase that the security level that the coming product should achieve is considered and decided.

In this phase, the developers need to think over the security for throughout the lifecycle. Thus, the developers must consider how much it will cost to achieve their target security level and how much budget should be allocated to each phase.

(2) Development Phase

In the development phase, the hardware and software are designed on the required specification which

outlines the purpose of the product. Vulnerability is likely missed and created into the product especially during the software design and coding phase. This guide provides the concrete vulnerability countermeasures in those stages. The guide also covers the security issues the manufacturer should take up from software development outsourcing, quality assurance, manufacturing at the factory and to shipment. The first priority at the development phase is to protect information assets. Table 3-2 shows the examples.

Table 3-2 Examples of Information Assets Embedded System Should Protect

Information Assets	Description
Contents	Multimedia data, such as sounds, images and movies (copyright management data when using commercial contents and private contents), content access log (note: it is important to protect content access log as well)
User Information	User's personal information (name/address/telephone number/date of birth/credit card numbers), credentials, access log, operation log
Product Information	Information on the product (model, ID（Identification, serial number）, machine credentials
Software Status Data	Software-specific status data (operational status, network usage status)
Software Configuration Data	Software-specific configuration data (operational settings, network settings, privilege settings, software versions)
Software	OS (Operating System), middleware, applications (also known as firmware)
Design Data, Internal Logic	Design data such as specifications developed at the planning/design phase

Besides information leak and data modification, security threats like the one where an embedded system is turned into a bot or used as a stepping stone to attack others, or the attack where the very service the embedded system provide is maliciously used (e.g. manipulation of home equipments) are also anticipated and countermeasures against these risks should be considered as well.

(3) Operation Phase

Depending on the way the embedded system is used, the developers should analyze its operation phase carefully. For example, an information home appliance is purchased by the users through the retailer. They sets up the appliance with their personal information and continuously add the private contents, such as sounds, music, images and movies into the appliance to enjoy them on a whim, ever expanding the information assets the embedded system should protect. During the period of the use of the appliance, the users may (1) set it up at home (the chance of other people directly operating it will be slim), (2) connect it to the home network, (3) connect it to the mobile terminals, such as cell phone, and (4) keep using it for a long period of time, like over a decade. These things should be also taken into account when considering the security at each phase.

Moreover, to respond to vulnerability that may be discovered in the future, the developers should implement a mechanism to update internal programs of the embedded system.

(4) Disposal Phase

The embedded system can be disposed at a timing when the users purchase a new one, stop using it or it is broken. The developers need to make sure that the users know how to remove the confidential information, such as personal information, when they dispose it. The same goes for when the memory unit built in the embedded system is broken and the users need to replace it or the users dispose the external storage devices (USB memory stick/SD memory card and HDD).

3.3. Security Levels

Up to this chapter, we have discussed threats and countermeasures to ensure the security of the embedded system considering its internal architecture and lifecycle. This, however, does not tell what exactly the developers need to do. One of the problems is the awareness gap between the management and the project team. For example, the management tends to brush off security as it is something the project team has to deal with on its own and does not give much policy guidance, and the project team, on the other hand, may put aside security issues due to not enough budget and the short development period. With the considerable scale of embedded systems commonly seen these days, however, it is quite difficult to assume that they are free of security risks, and the problems neglected in the upper process will reemerge in the lower processes with the increased difficulty and criticality. This can be solved only through the collaboration between the management and the project team where both of them properly understand security threats against embedded systems and work together. To visualize the current status of an organization's approach, this guide classified the security approaches into four phases based on the security policies the organization may employ (Table 3-3). IPA hopes this helps the project managers and system developers check out the security measures at the security level most close to the organization's target level to reacknowledge their current status and clarify the issues they have to work out. In the following sections, the possible management policies at each phase at each level are described. More concrete security measures required under each policy at each phase are given in Chapter 4.

Table 3-3 Security Levels at Each Phase of Lifecycle

	Management Policy	**Planning Policy**	**Development Policy**	**Operation Policy**	**Disposal Policy**
Level 4	Security effort is considered as an organizational issue. A security policy is drawn up and enforced, and an audit is also conducted.	Based on the organization's security policy, the security rules that can be evaluated objectively are established and enforced.		An vulnerability policy is drawn up as an organization and made public.	A disposal procedure that mitigates the security risks based on objective standards is available.
Level 3	Security effort is considered as an organizational issue. A security policy is drawn up and enforced.	Based on the organization's security policy, the security rules are established and enforced.		A vulnerability policy is drawn up as an organization.	A disposal procedure that mitigates the security risk is available.
Level 2	Security effort is left to the project managers and systems developers. Issues are dealt with separately at each project.	The security rules are established and managed by the project managers and system developers.		A vulnerability policy is drawn up for each product.	A disposal procedure is specified and written down in, for example, the specification documents.
Level 1	No security effort is done.	No security rules in place.		No assurance criteria for vulnerability.	No disposal procedure in place.

3.3.1. Management Policy

A management policy is an attitude of the management toward the security. The main issues here are that how well the management and the project team share their vision, draw up and enforce the policy. The security efforts at each level are summarized below.

- Level 1

  At this level, no security effort is done. This could be applied to the case where there is an awareness gap between the management and the project team, and as a result, the management tends to brush off security as it is something the project team has to deal with on its own and does not give much policy guidance, and the project team, on the other hand, may put aside security issues due to not enough budget and the short development period. The quality of the product strongly depends on whether or not it is kind of products that are susceptible to the security issues.

- Level 2

  At this level, security effort is relegated to the project managers and systems developers, and the security issues are dealt with separately by each project. This could be applied to the case where the management and the project team may consider that security is something both of them should work on together as an organization, yet, they do not know how and the issues are left to the project team in reality. The quality of the product differs per project and the know-how and experiences learnt in one project are unlikely shared with other projects.

- Level 3

  At this level, security effort is considered as an organizational issue, and a security policy is drawn up and enforced. This could be applied to the case where the management and the project team share the awareness of the security issues and document the policies, provide security education to the project team and share problems and issues faced by the project team. Since they have an organizational policy and effort, improvement of the product quality can be expected. On another front, they still have challenges to address; that they should review and make sure that the policy is adequate and the rules are properly followed based on some objective evaluation process.

- Level 4

  At this level, security effort is considered as an organizational issue, and a security policy is drawn up and enforced. In addition, an audit is also conducted. This could be applied to the case where the management and the project team share the awareness of the social responsibility they should assume over manufacturing and selling the product, implement the process to objectively evaluate their effort, and set up a new team and improve floors and factories to promote their effort. To achieve this level, the management and the project team must work together hard. Ultimately, this is the level this guide hopes that the organizations will aim and achieve.

3.3.2. Planning and Development Policy

A planning and development policy is an attitude toward security regarding planning and developing a product. The main issues here are that how well security rules are established and enforced in the development. The security efforts at each level are summarized below.

- Level 1

  At this level, no security rules are established. This could be applied to the case where there is no security standard and security risks are not discussed at the planning phase. The quality of the product strongly depends on the specification documents and the security problems tend not to emerge until the testing or operation phase. Moreover, they will be likely the types of the problems that have not been anticipated and may take very long time to solve.

- Level 2

  At this level, the security rules are established and managed by the project managers and system developers. This could be applied to the case where the project team is aware that the implementation of security measures and some method to evaluate them are necessary to solve the problems like vulnerability, and the possible security issues at the planning phase are well discussed. The quality of the product depends on the skills and knowledge of the project team, but it can be possible to eliminate common security risks through drawing up the specification documents.

- Level 3

  At this level, the security rules are established and enforced based on the organization's security policy. This could be applied to the case where the project team prioritizes what to protect based on the policy and establish security rules. They need to prioritize because it is not feasible to take care of everything even if there is some systematic method to look up vulnerability, implement and assess security. It is a given that the project team has a good security education but it can be expected to eliminate most of the security risks in the upper processes.

- Level 4

  At this level, the security rules that can be evaluated objectively are established and enforced based on the organization's security policy. This could be applied to the case where the project team has knowledge of the frameworks, such as JISEC (Japan Information Technology Security Evaluation and Certification Scheme) or JCMVP (Japan Cryptographic Module Validation Program) through their security education, establish the security rules that will meet the industrial standards, check out vulnerability information and evaluate the effectiveness of their security rules in collaboration with the expert. When something happens, with this level, the developer can have a formal path to investigate and report the causes of the problem and quickly solve it. Ultimately, this is the level this guide hopes that the organizations will aim and achieve.

3.3.3. Operation Policy

An operation policy is an attitude toward vulnerability response after the product is shipped. The main issue here is that how the developers respond to vulnerability to keep the product safe. The security efforts at each level are summarized below.

- Level 1

  At this level, no assurance criteria for vulnerability are established. This could be applied to the case where the developers do not share vulnerability information within the organization, do not let the users know of the voluntarily and wait till they do get complaint from the users. Then, they look into the matter and decide whether to leave it as a feature or to fix it depending on its seriousness.

- Level 2

  At this level, a vulnerability policy is drawn up for each product by the project team. This could be applied to the case where the project manager and system developers list up the vulnerabilities in the product and let the users know when they are serious. The users with low vulnerability awareness may just ignore security patches or software updates and keep using the vulnerable version, making the problem even worse. The vulnerability information is not effectively shared within the organization and the same vulnerability may show up again.

- Level 3

  At this level, a vulnerability policy is drawn up as an organization. This could be applied to the case where the developers keep tabs on the vulnerabilities, decide when and how to respond and make sure to let know the users if it is serious. On another front, they still have challenges to address; that they are not making use of vulnerability information available outside their company, and as a result, a known vulnerability that has been fixed in the products of other companies may show up in their product.

- Level 4

  At this level, a vulnerability policy is drawn up as an organization and made public as well. This could be applied to the case where the developers report the found vulnerability to the relevant organizations, such as JPCERT/CC, and utilize vulnerability information available outside their company. By reporting the vulnerability, the company can assess the vulnerability on the open verification environments or expect to have open source drivers and libraries fix. With this level, the developers can expect to improve the security of the embedded system by implementing the countermeasures against the known vulnerabilities. Ultimately, this is the level this guide hopes that the organizations will aim and achieve.

3.3.4. Disposal Policy

A disposal policy is an attitude toward security regarding how to protect the confidential information when disposing the product. The main issue here is that whether or not a disposal policy and procedure are established. The security efforts at each level are summarized below.

- Level 1

  At this level, no disposal procedure is specified. This could be applied to the case where whether the users keep it or dispose it is up to the users and nothing about disposal is mentioned in the specification documents.

- Level 2

  At this level, a disposal procedure is specified and written down in, for example, the specification documents. This could be applied to the case where the developers thinks it s mandatory to let the users know about the security risks and the recommended procedure when disposing the product since it will come to the end of its lifecycle eventually. Depending on the product, however, it may be infeasible if it requires a specialized tool and such to remove the confidential information stored on it.

- Level 3

  At this level, a disposal procedure that mitigates the security risk is available. This could be applied to the case where the product is equipped with a data destruction tool or the company offers a recycling system. These solutions can be a paid service.

- Level 4

  At this level, a disposal procedure that mitigates the security risks based on the objective standards is available. This could be applied to the case where the disposal is done in a way that NSA (National Security Agency) of the United States has established, or banning the disposed embedded equipment to reconnect the system by using a tracking technology. With this level, it is assured that the product will not contain any personal information or confidential information. Ultimately, this is the level this guide hopes that the organizations will aim and achieve.

4. Security Measures at Each Phase

4.1. Management

4.1.1. Security Rules

In providing embedded systems, the developers should not settle on ad-hoc security. The developers need to establish a security policy and define the security rules, what must be done and what must not be done, as an organization.

For example, the developers need to make the security rules regarding procurement when buying the modules that consist of embedded systems from external vendors. If the developers develop the modules in-house, they need to establish the security rules on how to implement security measures and features or how to establish a project teams and developmental environment that ensure the reliability of the processes in during the development and manufacturing. In addition, to provide a customer support service and protect the personal information of the users, a policy to establish a security management system is mandatory. The developers can consult ISMS (ISO/IEC27001) as a reference for the security rules. Meanwhile, below are examples of concrete policies and rules organized in the way that shows which one corresponds to which phase of the lifecycle presented in Figure 3-2.

【Common throughout the Lifecycle】

- Information Security Policy

  Draw up the organizational policy on information security, according to its business requirements, relevant laws and regulations.

- Information Security Management Policy

  Establish the rules to appoint a chief information security officer and organize a system to promote and manage security within the organization.

- Human Resource Management Policy

  Draw up a non disclosure contract and establish the rules on recruiting and security education to hire appropriate personnel and protect the confidential information from disclosed. These rules must be applied to temporary workers and contractors who are involved in development of embedded systems.

【Designing / Development】

- Development Policy

  Establish the rules, for example, to ban unauthorized physical access or to prevent malware from infecting computers to ensure security of the development environment.

- Design Policy

  Establish the rules to decide the security mechanisms required to prevent, detect and recover from security incidents when designing and implementing.

・ Procurement Policy

Establish the rules that make sure that the hardware and software purchased from external vendors have the necessary security features and that they are implemented properly and reasonably certifiable in the eye of the outside party.

【Operation】

・ Operation Policy

If vulnerability is found in the embedded systems already in hands of the users, the developers need to prepare a security patch and let the users know about the vulnerability and patch. Establish a incident response system and the procedure, and the rules to appropriately handle the users' personal information, such as contact information.

【Disposal】

・ Disposal Policy After Collecting Disused Products

If the products that may store personal information are also covered in the collection, establish the rules in accordance with the Act on the Protection of Personal Information.

Level	**Description**
Level 1	・ No security rules are established.
Level 2	・ Security rules are established voluntarily by the developer. ・ Security rules are established by the project manager and system developers.
Level 3	・ Security rules are established as an organization and documented.
Level 4	・ Security rules are established as an organization and documented. An audit to evaluate the compliance to the rules is also in place.

4.1.2. Security Education

Embedded systems have evolved into a system that consists of advanced hardware and a collection of sophisticated applications. This has increased the risk of security flaws (vulnerabilities) in the embedded systems that may let the users do something they are not supposed to do, such as unauthorized operations and access to the data. The vulnerability may exist in the source codes written by the developers themselves or outsourced vendors, or basic hardware and software.

Attackers and researchers have been in fierce competition to find a way to exploit the vulnerabilities and new techniques are found daily. It is good for the developers to provide security education on the following issues to foster the system developers who are strong in the security area.

- Known Vulnerabilities

  A vulnerability found in the software is often a known one. For example, the JVN vulnerability countermeasure information[4] provide the case studies on a variety of vulnerabilities, such as cross-site scripting, cross-site request forgery, directory traversal, SQL injection in a management GUI for the embedded systems. The developers need to learn from the case studies available on the Internet and make sure that they do not have the known vulnerabilities in their system.

- Secure Programming

  Sometimes security holes are created during the implementation. For example, if a library that provides authentication or encryption is not properly implemented, an attacker may be able to bypass authentication or break the encryption. Something that the system does not expect to handle, say, too long strings or unsupported character sets may cause an undesirable behavior of the system as well. For more information on secure programming, check out Secure Programming Course[5] or guidance is also found on the e-Learning Websites or books on the subject.

- Security Testing

  It is critical to perform security testing at the design and implementation process to make sure that the security mechanisms are properly implemented. To be specific, the following must be done. (1) To review the design documents to make sure that necessary security measures are put in place, (2) to review the source codes to make sure that they do not have vulnerability, (3) to perform system tests on the developed system to make sure that they indeed do not have vulnerability. More information on security testing can be found on the e-Learning Websites or books.

  For security verification of source codes, there are specialized tools for that purpose. There is also a verification tool that systematically analyzes an embedded system with a network connectivity for the known TCP/IP vulnerabilities[6]. The embedded systems may use commonly attacked Internet protocols (such as http, sip and dns). In that case, the developers had better be aware of verification tools for the vulnerabilities in these protocols as well. For example, IPA offers a verification tool for SIP[7], a protocol used to offer IP phone services. As for the commercial tools, there are Nessus[8] (vulnerability scanning tool), nmap[9] (port scanning tool), Metasploit[10] (vulnerability scanning tool using attacking code), nikto2[11] (Web server security

assessment tool). In addition, the use of a fuzzing tool, which analyses and looks for the system's unexpected response caused by the erroneous input, such as Protos[12], can be also effective. Having knowledge on these tools is also encouraged.

Level	Description
Level 1	・ No security education is done.
Level 2	・ A voluntary working session is held. ・ The project manager and system developers acquire the knowledge or organize study sessions as needed.
Level 3	・ Security education is provided as part of daily work. ・ Attending security seminars and taking security training are encouraged.
Level 4	・ Security education is provided as part of daily work. ・ Taking security training with an expert is required.

---Coffee Break---

Terms related to vulnerability are mostly foreign-made. In addition to their difficulty to understand, the range of the hardware and software components vulnerabilities exist is overwhelmingly wide. We, however, now come to the point where we must face and understand them. We used to be safe when we were expected to achieved some limited functions using limited resources for the limited usage environments. As systems have been more and more networked, sophisticated, serviceable and large-scale, however, we have no choice but use the hardware and software components developed by someone else. We must accept and deal with the vulnerabilities we have no control over.

If vulnerability exists in the product and is exploitable, the product we have developed is a sitting duck for an attacker. Even if the product has developed for specific use and contains no confidential information, that does not change the situation at all. The attacker may not be interested in our product itself, but the attacker can exploit the vulnerability and use its processing power to attack other targets, possibly making our product an accomplice in a crime.

Even the best security system cannot solve every security problem. Let's see TPM (Trusted Platform Module) as an example. TPM is versatile where it can do things, such as detect software modification, check the authenticity of devices and provide encryption. TPM, however, is just a system itself. At a security conference BLACK HAT 2007, a presentation called "TPMkit: Breaking the Legend of Trusted Computing (TC [TPM]) and Vista (BitLocker)" was scheduled but cancelled. No reason was given. Despite its intriguing claims, the tool has only 4 hits on Google Japan. What does it mean? It is important to have critical eye.

First, we should understand what are used in our product, and then, see if they have vulnerabilities and how the vulnerabilities have been exploited. Do not limit the search within Japan but look overseas as well. We will see there are more risks than we anticipate. We may not be able to understand what we find in our research much, let alone judge if the vulnerability indeed affects our products.

That is exactly why we need to learn and acquire knowledge.

4.1.3. Collecting Security Information

To develop and sell an embedded system, it is necessary to collect and respond to security information related to embedded systems throughout the system's lifecycle. Especially for open source software, the developers must realize they themselves are solely responsible to collect the necessary security information. At the development phase, the developers should make sure that the hardware and software they are going to use have no security problem. The information the developers should collect are as follows.

- Vulnerability Information

  For example, when using a popular software used in Japan, such as Linux, to develop an embedded system, it is critical for the developers to know all the latest vulnerability information on OS and middleware provided by their development communities. The developers can also collect the information on vulnerability found and officially reported by searching a vulnerability countermeasure information database JVN iPedia[13]. In addition, there is MyJVN [14], a filtered vulnerability countermeasure information tool, which provides the features to filter and auto-recollect the vulnerability information, and a checklist of vulnerability countermeasures to enable the users to collect the vulnerability information only relevant to them.
  JPCERT/CC is a designated organization to collect and distribute computer security information in Japan and support security incident response. It works under the Framework for Vulnerability Information Handling"[15] where JPCERT/CC provides the vulnerability information yet disclosed to the public to the information system developers on a need-to-know basis to avoid misuse of the vulnerability information and minimize the possibility of causing harm. To be among the first to obtain the vulnerability information, it is useful to participate in the framework.
  Online news sites also cover the security issues about IT products. For example, SecurityFocus[16] gives the information on vulnerability, attacking methods, countermeasures and security tools.
  As for open source software, the developers should check out the product's update information released by the original developer.

- Trends in Encryption Standards Used in Authentication and Data Protection

  Every day, researchers and crackers worldwide are trying to crack the encryption and authentication technology. In Japan, CRYPTREC (Cryptography Research and Evaluation Committees) works on a project to conduct a research and create a list of encryption technologies that are safe and well implemented based on the objective evaluation. When developing an embedded system using encryption technologies for authentication and data protection in Japan, the developers should refer to the report[17] from CRYPTREC. In other countries, since each country has its own rule, the developers need to refer to the encryption standards set in the country.

Level	Description
Level 1	・ No security information is collected.
Level 2	・ Security information is collected by the project manager and system developers as needed.
Level 3	・ There exists a system where security information is collected as an organization and disseminated to those involved in system development.
Level 4	・ Is participating in the Vulnerability Information Handling and utilizing vulnerability information.

4.2. Planning Phase

4.2.1. Budgeting

The developers need to ensure budget for security throughout the embedded system's lifecycle (planning, development, operation and disposal). Especially, security relevant software (such as OS and encryption libraries) requires continuous budget for updating in addition to the initial cost at the development phase.

Since security problems are difficult to predict in advance and often come out into the open sometime after the product release, it is desirable for the developers to have an organization-wide, continuous budget to avoid the risks besides the security budget to develop the product. To carry out thorough security awareness throughout the organization including the management, the budget for the following issues should be included in the project approval and financial documents based on the project's development plan.

- At the time of development
  - Research cost (books, seminars, trainings)
  - Procurement cost (hardware, software, certificates for testing and operation phase)
  - Quality control management cost (cost to build a revision management system)
  - Cost to build test environments

- At the time of operation
  - Research cost (check and update hardware, software and certificates used in the product)
  - Maintenance cost (maintain and update the test environments, hardware, software and certificates)
  - Operation cost of the revision management system[18]
  - Expenditure to respond to any security problems that might emerge (it can be per product or for all products as a whole. Consider to provide the users and administrators some options to update the product easily depending on their use environment, such as update through the Internet or some media.

Level	Description
Level 1	• No budget for security issues is ensured.
Level 2	• Budget is ensured when the project manager requires. • Budget is considered when the project manager and system developers require.
Level 3	• There exists a system where a certain amount of budget is allocated to ensure security as part of the development process.
Level 4	• Budget is allocated to set up a specialized security team and put it on work.

4.2.2. Selecting Development Platform

Here, the development platform is the base hardware and software that compose a product to be developed. When developing an embedded system, the developers select the hardware components, such as CPU and memory, and OS and various software applications to control them to implement the planned functions and capability. The basic development technique is to select all components from scratch and build a customized development platform.

In these days, however, a development technique, in which the developers use a reference board (motherboard) shipped by the chipset makers who provide CPU and other devices as a base and add their own customization to it, has become common to shorten the development period and reduce cost. These reference boards make debugging during development easier and have a good scalability, but security issues are often not much concerned. When designing the hardware, the developers may use the devices and circuit design as they are to maintain the performance reliability, but it is quite dangerous from the security perspective. For example, a physical attack has been reported where the confidential information was stolen through probing when the information was transmitted in plain text on the data bus on the motherboard. If the developers use a reference board as reference when designing, they must take into account such physical attacks and implement countermeasures.

When selecting the hardware, the developers need to keep the possible physical attacks and countermeasures against them in mind as follows.

- When the confidential information is to be transmitted on the data bus on the motherboard, make the circuit architecture difficult for an attacker to intercept the data. When deciding the system architecture using a reference board, check if the circuit architecture can be used as it is.
- For a device that is to handle confidential information, choose a BGA (ball grid array) package device or something that could make probing difficult.

Next, let’s think about the software. The software for an embedded system is selected based on the hardware architecture. A real-time OS and control driver software are selected depending on CPU and the devices. Besides OS and the drivers, off-the-shelf, software packages that provide a specific function are also available. In recent years, as embedded systems have become multifunctional and high-capacity, more developers tend to adopt open source software, such as Linux.

Previously, embedded systems with a network connectivity are equipped with the protocol stack for only IPv4, but those to be developed in the future will need to support IPv6 as well.

In each case, it is important to check out the security policy of the vendors whose embedded software package the developers planned to use for their product in advance. Here, the security policy means the level of support that can be judged from the software update cycle or the length of support period, whether the vendor provides a security patch or update in response to vulnerability information, in addition to the ordinary bug fixing and version ups. For those that have not been time-proven, such as the protocol stack for IPv6, the developers should check and double check to be sure.

Threat information is available from a vulnerability countermeasure information database JVN iPedia[13], as it is mentioned in 4.1.3 Collecting Security Information. To determine how an embedded software package vendor has

been responding to published vulnerability information could be one of benchmarks. The developers may find out how secure the software package is from other company's case study.

When building a unique development platform without using embedded software packages, the developers need to pay great attention to vulnerability information as pointed out in the above section. It is critical to monitor what is going on with all hardware and software components and provide a security patch or update timely.

When selecting the software, be careful with the following points.

- When using an embedded software package, check out the vendor's vulnerability support system in advance.
- When building a unique development platform, do not forget that the developers need to collect vulnerability information and implement countermeasures on their own.

Level	Description
Level 1	・ Platform is selected based on cost and delivery period. Security is not considered much.
Level 2	・ Known vulnerabilities at the time of development are taken care of. ・ After shipment, nothing is done. ・ There is no unit or person in charge that selects the development platform. ・ The level of vulnerability awareness differs depending on each project manager and system developer, thus no unified effort as an organization exists.
Level 3	・ When using embedded software packages, monitor information on security patches for them and update released by their vendor. ・ Security patches and updates are provided after shipment as well. ・ Vulnerability awareness is high in the units that manage the same product and unified effort is being done.
Level 4	・ Vulnerability countermeasures are done across the organization. ・ Security patches and updates are provided after shipment as well. ・ A vulnerability support system of the vendors is checked out in advance. ・ Devices and circuit architecture are selected considering security as well. ・ To improve security of products, efforts like building a unique platform or requesting the vendors for improvement are being done.

---Coffee Break---

IPv6's 128 bit address provides an enormous number of total IP addresses (340 undecillion addresses) and promises flexible end-to-end communication by allocating an IP address to every single embedded system. It also has various advantages, such as helping automation of complicated IP settings and standard support for cryptographic communication and mobility by expanding the header functions. A huge address space of IPv6 means that an attacking method like exhaustive IP address scanning will become very difficult to perform.

On the other hand, it is possible that these characteristics of IPv6 may turn out to be the source of new risks, and become the disadvantages compared to the current situation with IPv4. Many coming embedded systems will support IPv6. The developers should realize that security requirements for embedded systems, which are not used to be attacked via the Internet except some networked ones, will become higher in the future.

4.3. Development Phase

4.3.1. Designing

In a general process of an embedded system, there is a design process in its early development phase in which functional specifications and capabilities are decided. The hardware and software architecture to support the functional specifications are also decided at this phase and then each function is discussed in detail. It could be possibly rare that security effort described in this guide is actually performed in real development proceedings.

In recent years, however, an increased number of embedded systems have a network connectivity and the developers must realize that the threats of becoming a target by attackers are real. If a product is shipped with vulnerability and it causes a problem later, it will cost the developers a big money to recall and fix the problem. In the case of a cell phone incident reported a few years ago, it was said that the total cost was over 10 billion yen. To avoid such situation, it is important to examine the security requirements for a planned embedded system in the design process during the so-called upper process and consider what kind of security countermeasures should be implemented.

Based on the internal architecture of an embedded system described in 3.1.1, some examples of security requirements that should be discussed in the design process are listed below. Some of them may not be considered as a security requirement depending on how the product is used. In that case, the developers should select only what they need based on the purpose and expected user base of the planned embedded system.

Also, it is necessary to implement the ways that support the users to safely use the system. For example, the security features mentioned in this section (like encryption of transmission or countermeasures against attacks) are enabled when the user uses the system at the default settings or follow the procedures suggested in the manual,or issuing a warnings when the security features are disabled.

- Protection of Confidential Information

  When an embedded system with a network connectivity is attacked, it is necessary to prevent the confidential information stored in the system, such as personal information, from being disclosed. The confidential information includes information assets that may harm the user. (*for more details on information assets, refer to Table 3-2 Examples of Information Assets Embedded System Should Protect) Depending on the level of the confidentiality of the information assets, the logical or physical location of data, encryption method, strength of encryption are decided.

  Countermeasures:

  - To prevent spoofing when transmission takes place, fortify user and server authentication (use password or biometric authentication for within the embedded system and for authentication via network, employ those that is not vulnerable to replay attacks).
  - To prevent data leak via network, bus and memory units, select the encryption algorithms appropriate for the confidentiality level.
  - Consider using encrypted memory and hard disk.
  - Enable the use of USB memory stick and IC card and do not allow the internal non-volatile memory to store sensitive information.
  - Implement an import/export feature for personal information to prevent information leak in casethe

embedded system is sent for repair to an external party.

- Consider implementing a way to manage the user information on the server via network.

- System Failure Recovery

When an embedded system becomes under denial of service (DoS) attacks, it is necessary to prevent system failure, such as freezing and rebooting, caused by the attacks.

Countermeasures:

- Implement a mechanism to monitor itself and detect failure, recover and shut down.
- Implement a mechanism to alert the user about failure, on display or via network.

- Alerting

In the case where an embedded system is attacked, for example, data is modified or destroyed, privilege is escalated, or it is under DoS attacks, it is necessary to have a mechanism to warn the user about the attack. A mechanism to let the user know when the user performs erroneous operation or configure incorrect settings should be also implemented.

Countermeasures:

- Implement a user interface to send an alert to the user when the system is attacked.
- Clearly specify emergency response procedures, such as cutting off from the network, in the alert message.
- Clearly specify the detailed explanation on each alert message in the manual and user's guides.
- Require password to change the system settings.
- Consider to put restrictions on changing port numbers and port usage.
- Implement a user interface to send an alert to the user when the user configures network settings that may put the system in danger.

- Logging

In the case where an embedded system is attacked, for example, data is modified or destroyed, privilege is escalated, or it is under DoS attacks via network, it is necessary to have a mechanism to log suspicious login attempts and events within the system.

Countermeasure:

- Determine whether a logging feature for data transmitted and received is necessary. Log may include the source IP address, destination IP address, port number, data size, date/time.
- Determine whether a logging feature for users' login records. Log may include user ID, number of logins, number of login errors, date/time.
- Determine how long to keep each log and decide the data size and memory configuration.
- When managing logs online, decide what protocol and data format should be used.

- Services

It is necessary to prevent an attacker from accessing the embedded system and obtainig the system information, such as software configuration, through the special-purpose features that are normally off-limit

to the user, for example, the testing function used in the factory, a maintenance feature for service crews, and a sales demo tool. In addition, unauthorized access through the debug connectors used during development or communication connectors used by service crews should be prevented as well.

Countermeasures:

- Require a set of procedures to access each special-purpose feature, making it difficult for the user to use it.
- Remove all features and function that are unnecessary before shipment.
- Make sure that how to access the special-purpose features is not disclosed on the Internet.
- Do not implement a debug connector for mass production hardware or remove the connection pattern.
- Customize the shape and way to implement for the communication connectors used by service crews, making it difficult to access.
- Communication connectors used by service crews can be used to read and update the information stored within the embedded system. Secure the communicate with the microcomputer by requiring a set of procedures to proceed.
- Make it harder to analyze the software.

・ Physical Countermeasures for Embedded System Hardware

It is necessary to implement anti-tamper techniques to prevent malicious attempts, such as, opening the computer case forcefully, unauthorized access to the devices on the motherboard, probing and modification of data in volatile memory.

Countermeasures:

- Implement a mechanism to automatically delete the confidential information when the system detects the case has been opened forcefully or hard disks or memory is removed.
- Implement a mechanism to send an alert online when the system detects the case has been opened forcefully and hard disks or memory is removed.
- For those that handle the confidential information, choose a BGA (ball grid array) package device or something that could make probing difficult.
- After mounting the devices, coat chips and devices physically or chemically to prevent probing.

・ Disposal

When the product is given over from one user to another or disposed, the confidential information stored in the hardware may be disclosed. To prevent this, it is necessary to consider to maintain data on the network instead of storing in the product. For the data that must be stored in the product, provide a way to initialize or destroy the confidential data.

Countermeasures:

- Make the hardware configuration supportive to categorize the data into two, confidential data and other data, and store them separately to enable partial removal of the data when needed.
- Implement a user interface that allows the user to easily remove and initialize the data.
- Consider a mechanism to physically remove sectors on the hard disk.

- Enable the use of USB memory stick and IC card and do not allow the internal non-volatile memory to store the confidential information.

- Countermeasures against Being Used as a Stepping Stone

It is necessary to prevent the embedded system with a network connectivity from being attacked and compromised to attack others. It is also possible that an attacker tries to harm others by invoking the embedded system's abnormal behaviors via network to overflow the traffic and that should be prevented.

Countermeasures:

- Fix vulnerabilities so that an attacker cannot exploit them.
- Implement a mechanism to detect attacks via network and disable them. Below are some benchmarks useful to detect attacks.
  - Receiving too many packets in a certain period of time
  - Too many failed login attempts in a certain period of time
  - Receiving unexpectedly large packet data
  - Access to unused ports

- Countermeasure against Side Channel Attack

Recently, an attacking method called side channel attack has come to light. A side channel attack is an attack based on the information obtained from the physical implementation of encryption algorithms, such as timing information, power consumption and error messages, and exploit them to break into the target system.

Typical side channel attacks are as follows.

Table 4-1 Examples of Typical Side Channel Attacks

Attacking Method	Description
Probe Attack	Open the LSI package and insert ping into the chip's data bus to monitor and analyze the change in data on the buses.
Power Monitoring Attack	Similar to timing attacks, analyze power consumption by the hardware during computation to make an educated guess on the confidential information. There are some variations such as simple power analysis (SPA) and differential power analysis (DPA).
Electromagnetic Attack	Observe leaked electromagnetic radiation to analyze the hardware's operation. Like power consumption, it will provide extra source of information.
Differential Fault Analysis	Intentionally introduce a failure to the module, have the module operate and analyze the erroneous outcome to gain an idea on the confidential information. To make the module fail, an attacker may subject the embedded system to unsupported supply voltage or current, overclocking, strong electromagnetic fields, radiation or strong light.
Timing Attack	Analyze the time taken to execute cryptographic algorithms to gain idea on the sensitive information. For example, when performing a Montgomery multiplication, it may or may not perform modulo operations at the end of calculation depending on the value of the interim outcome of the operation. By repeatedly performing Montgomery multiplications with different input values and measuring the time taken to process the calculation, an attacker may obtain the information that may provide hints to assume the confidential information.

Countermeasures:

The countermeasures against each attacking method are summarized below.

- For the LSI that processes the confidential information, use those that have a protective layer to block

observation from the outside to prevent probe attacks. The other options will be to surround the circuit with a metallic layer, monitor the current on it and destroy confidential information when the system detects tampering, or destroy the chip when a protective layer is removed (anti-tampering mechanism).

- To prevent power monitoring and electromagnetic attacks, make sure that there will be no difference in the measurements regardless of the content of the calculation operation, just like to prevent timing attacks.
- Implement a mechanism to detect a failure and destroy the confidential information when detecting it to prevent differential fault attacks.
- Insert dummy routines to the algorithms so that there will be no difference in the measurements regardless of the content of the calculation operation to prevent timing attacks.

As information home appliances become more advanced and popular, the applications and contents that used to be for the PCs only are now also available the for information home appliances. Some information home appliances have a capability to upload the contents to the Internet or share files like a server. This has introduced a risk of being attacked through the Internet and need for implementing the countermeasures to the information home appliances as well. Below are some examples of the features that may be needed for the information home appliances that support common Internet applications.

- Countermeasure against Malicious Programs

Depending on the functions the embedded system provides and risks posed by the services it uses, it is necessary to implement a mechanism to detect, remove or correct malicious programs and modification attempts hidden in the files, the memory and transmission data, and a way to update the pattern file for the detecting mechanism.

Countermeasures:

- Implement a mechanism to detect malicious programs on the embedded system or utilize an equivalent service provided on the Internet.
- When implementing a mechanism to detect malicious programs on the embedded system, make sure that it has a way to update the detecting mechanism and pattern files.
- Implement a mechanism to send an alert to the user when a malicious program or modification attempt is detected.

Level	Description
Level 1	・ No security is taken into account at the design phase and it is not included in the specification documents.
Level 2	・ Consideration on security at the design phase is up to the project manager and system developers. ・ No organizational policy is established and an audit is not done nor addressed.
Level 3	・ An organizational security policy that should be followed at the design phase established. ・ The design process is preceded following the policy. ・ An audit is not done.
Level 4	・ An organizational security policy that should be followed at the design phase established. ・ The design process is preceded following the policy and there is an audit process to review the security measures.

---Coffee Break---

In one of researches on differential power analysis (DPA), the researchers are studying on circuit configurations and algorithms that will consume the same power regardless of the content of cryptographic operations to nullify the correlation between the confidential information and power consumption. However, the countermeasures recommended at the academic conferences are too expensive or infeasible with the current LSI designs. In reality, there is no established, working solution for the mass produced LSI design yet. Even if there is no countermeasure against power monitoring attacks, it can be possible to reduce the power consumption to quite low by using nanometer chip. Consequently, it will make power monitoring attacks harder.

Also, it will be effective for mitigating power monitoring attacks to design a separate circuit dedicated to cryptographic operations since it is easier for an attacker to analyze power consumption when CPU is used for cryptographic operation, or to change the cryptographic key often.

What is important here is to realize that when a planned embedded system is to perform cryptographic operation, it will be nonsense to worry about side channel attacks too much and unnecessarily increase the cost but do not down play and find a good balance between the effectiveness of available security measures and feasibility of implementation and put it into the design.

4.3.2. Software Implementation

The security problems in software programs can be a bug that arises spontaneously during operation or one that the behaviors specific to an embedded system are analyzed through direct or indirect attacks and turned into a vulnerability. To counter direct and indirect attacks, it is recommended that the developers adopt anti-tamper implementation and secure programming when coding the software. Anti-tamper technology aims to protect the software and data from attacks that try to directly analyze an embedded system through the data in the memory, source code itself, or data transmitted on the bus. Examples of the countermeasure are software hardening and bus data encryption. On the other hand, secure programming aims to protect the software and data from attacks that try to indirectly analyze an embedded system's responses to unsupported packets or commands. Examples of the countermeasure are packet filtering or a countermeasure against buffer overflow.

When implementing security, the developers should note that there are tradeoffs between security and those such as software's capability or debugging efficiency.

The software implementation processes have two general phases: source coding and its review. There are various coding methods and review techniques to improve the quality of software. It is also possible to enhance the efficiency and accuracy of reviews by utilizing tools for analyzing source code. Because what is most appropriate differs depending on the size of software, it is difficult to suggest the best technique and method. Here, some researches on vulnerability countermeasures during software implementation are introduced. The developers should evaluate the security level of their organization carefully based on the description introduced later in this guide and pick up the advices accordingly to their operation. Meanwhile, they should know that learning coding methods and review technique require expertise, and the organization must work on it as an organization through security education or other efforts.

Study Report 1:

Secure Programming Course[19] on IPA Website can be of help to counter buffer overflow or format string attacks. Its contents are not specialized for embedded systems, but Chapter for C/C++ is especially useful for embedded system software development.

Study Report 2:

IPA's research project "Toward Establishment of Methods to Comprehend and Review Software Source Code[20]" can be of help. It is still an ongoing project but a draft version is available for the public. It introduces methodologies and techniques to analyze, evaluate and review software code. It is distinctive where an expert is assigned as the technical manager to each software programming language or OS, and source code analysis is the main focus.

Level	Description
Level 1	・ No security coding is done. ・ No source code review is done. ・ Source code reviews are done but security is out of scope.
Level 2	・ Coding policy and review technique to prevent vulnerability depend on the voluntary effort by system developers or some project members. ・ No uniformed effort is done even in the same department. ・ Because it is done based on the knowledge and experiences of the project manager and system developers, there are no clear criteria.
Level 3	・ There is an organizational policy and rules about coding and reviews that should be followed in software implementation to prevent vulnerability. ・ System developers write source code following the rules. ・ An audit of the compliance with the rules within or across the departments is included in the development process.
Level 4	・ In addition to the efforts at the level 3, there is a specialized security team in the organization and it gives supports for secure cording and reviews. ・ The security team is also involved with the rule making and keeps revising the rules based on the information available on bugs, vulnerabilities, incidents and countermeasures.

---Coffee Break---

What is it that you do satisfy all the requirements yet write codes with security holes? Let's take a program with a cryptographic function as an example and look into the issue.

Many software now support encryption to protect data. From the attacker's perspective, encryption gives a clue that the data that requires encryption mean they are valuable information, and it is possible for the attacker to fish out sensitive information from hundreds megabytes of data by analyzing memory access when cryptographic operation is performed. It may be even possible to recover data encryption keys during the act.

How does the attacker find out where the encryption engine resides? One method is to focus attention on a characteristics that AES or DES repeats a particular process for predefined times and to analyze the source code looking for loops. It may not be able to pinpoint where the encryption engine is but does shorten the time it takes to analyze. Should we call it simply a bug when sensitive information is disclosed due to such an attack? Is there any way to stop this from happening?

Unfortunately, all we can do now is to make it a little harder for an attacker to pull it off. How much harder an embedded system should be? As the case now stands, they must judge on their own depending on how valuable the data they need to protect are and how much money they could spend to protect them.

4.3.3. Outsourcing

Due to the ever-growing scale of embedded systems and cost to develop them, some projects are outsourced to external parties. What the developers need to do to prevent vulnerabilities has been descried up to this section, and the developers must require the same security effort from the outsourcees. In reality, however, it would be difficult to apply the same level of awareness and rules due to the difference in organizational climate and culture. In addition, in a large-scale development environment where a number of people involve, possible communication barrier should be taken care of.

Especially when software development is outsourced, the risk of vulnerability becomes higher since the development process is not always visible and controllable. There is also the risks that the rules on coding and reviews are not enforced strictly and human errors can be missed The developers must take into consideration that someone who is not well skilled may be assigned to design the system depending on the outsourcee's resource.

There are various cases of outsourcing. Depending on the product's architecture and cost they can allocate for, the developers may manufacture hardware in-house and outsource only software development, outsource only part of software development, or employ some other style. Practicing the countermeasure introduced in this guide will be effective against vulnerabilities and the developers should employ accordingly to their outsourcing policy.

Outsourcees will get involved with the development in various ways. It is necessary to require security effort from the external parties based on the developers' internal security policy. Below are some pointes to keep in mind when outsourcing.

- When outsourcing the design of the basis of hardware or software, provide the design rules and criteria to the outsourcee and clarify each other's role and responsibility in the project not leaving all decision making to the outsourcee.
- When outsourcing software implementation, clarify the coding rules and review method in advance and include them in terms and conditions of quotation.
- To make sure the security effort is properly performed as required, hold a joint review and require to submit the design documents. By reviewing the documents and auditing the processes, establish a system to check and ask for improvement.
- Discuss a security assessment and debugging method in advance, and the cost for assessment tools and manpower.
- Define the acceptance conditions regarding security and require to submit a security specific document.
- Consider a communication method and environment with the outsourcee and enable feasible enforcement of security rules.
- Make clear the scope of responsibilities over the security matters and clearly written in the contract.

Level	Description
Level 1	・ No vulnerability countermeasure is taken into account when outsourcing. ・ Even if the project has security features, the same outsourcing policy is applied.
Level 2	・ Vulnerability countermeasure in the case of outsourcing is up to the voluntary effort by system developers or some project members. ・ No uniformed effort is done even in the same department. ・ Because it is done based on the knowledge and experiences of the project manager and system developers, there are no clear criteria.
Level 3	・ A contract for software development outsourcing requires an organizational security effort within the outsourced party to prevent vulnerability. ・ An audit of the compliance with the contract condition within or across the departments is included in the development process.
Level 4	・ In addition to the efforts at the level 3, there is a security team in the organization and it gives advice to the outsourced entity.

4.3.4. Security Assessment Test and Debugging

The attacks to disclose, modify or destroy data to be protected in an embedded system are executed via logical or physical interfaces to the information assets. Examples of attacks against logical interfaces are buffer overflow attacks that send the data the targeted software does not expect and support, or injection attacks that send a specially-crafted data embedded with malicious command and induce erratic software behavior.

Examples of attacks against physical interfaces are reverse engineering attack that analyze the software by connecting a debugger to the debug connector or side channel attacks that measure the changes in the system's physical quantity during operation, such as power consumption, and analyze the measurements to find out sensitive information like an encryption key. Security assessment test involves performing a security assessment based on the security requirements defined in 4.3.1 Designing and checking if the software is indeed safe against the various attacks mentioned above. When performing a test, it will be effective to use the security assessment tools introduced in 4.1.2 Security Education. Major attacking methods and advice on what should be checked for each attack are summarized below.

- Injection Attack

  Injection attack exploits control codes and inserts an unexpected command when the software builds a command string for the DB server or shell to execute. It is important to make sure that control codes in the command strings sent to the DB server and shell are appropriately escaped. .If the software offers a Web interface for management purpose, use the injection detection tool[21] provided by IPA.

- Buffer Overflow Attack

  Buffer overflow attack involves sending a maliciously-crafted program and executing it in the memory (including stack). To check if the software is indeed protected from buffer overflow attacks, look up the event where a data larger than the buffer size had been inputted and see if the software processed it properly as it was supposed to.

- DoS Attack

  DoS attack renders the server software within embedded systems incapable of providing their services. A DoS condition occurs when the server programs abnormally end, go into infinite loops or deplete recourses. Make sure that the software is capable to detect and avoid such conditions. As for known vulnerabilities in TCP/IP, use the vulnerability assessment tool[6] provided by IPA.

- Reverse Engineering Attack

  Reverse engineering attack involves analyzing hardware or software and studying about the products, such as its architecture, manufacturing technology and operating principles. In the software field, reverse engineering generally suggests a technique where an attacker statically analyzes binary executable files and finds out the product's design and implementation with a disassembler or dynamically with a debugger. Reverse engineering is one of the techniques to find vulnerability and very useful in that regard, but it also poses a risk that an attacker finds and exploits vulnerability unknown to the public and executes attacks if given a

chance to perform reverse engineering. By adopting software hardening technology, it is possible to make it harder for an attacker to discover vulnerability.

- Abuse of Service Ports

An attacker may connect a debugger to embedded systems and analyzed the behavior of the systems. Make sure that a connector used in debugging or maintenance mode for developers or service clues is off-limit for the users.

- Side Channel Attack

Side channel attack involves observing and analyzing the behavior of an embedded system to assume or find out sensitive information. Make sure that important, sensitive information is not easily analyzed.

- Format String Attack

Format string attack involves sending malicious data that the targeted software does not expect and support, in terms of the type or size of the data, and having the software execute arbitrary command. Check input data and see if the software stops processing it when the input contains unexpected, suspicious data.

Level	Description
Level 1	• No security testing is done.
Level 2	• Security test is planned based on the knowledge and experience of system developers • The quality of the security testing depends on system developers' knowledge.
Level 3	• There is an organizational security test plan and evaluation is done based on the organization's criteria. • If other device is connected, test the connectivity with the device,
Level 4	• Following the security the test plan drawn up as an organization, the internal security team reviews it. • In something is added, updated or delete after shipment, check the connectivity with the new device.

4.3.5. User Guide

Vulnerability may be found after an embedded system is in the hands of the user. The developers need to provide what to do when it is found beforehand, for example, putting that information in the user guide. What information should be given in the user guide is listed below.

- Guidance to use the product securely

  In the case that the embedded system has a network capability, it is desired to give instructions on how to protect data stored inside the system from an attacker (for example, how to set a password). It is also good to let the user know of possible consequences if he or she does not put up security measures (for example, files on the system may be deleted thorough unauthorized access).

- Trouble Shooting

  The information on what to do when the user faces some kind of trouble in using an embedded system (such as, shutting it down and rebooting it) should be provided. In the case that the developers provide software update or customer support to solve the trouble, the procedures to receive those services need to be provided as well. If the embedded system has a password protection feature, what to do when the user forgets the password and the possible impact of initializing the password had better be mentioned (for example, some contents may become inaccessible, the settings may be lost, services may become unavailable due to deletion of the certificate, a theft may initialize the password and hijack the system).

  In addition, an embedded system's security setting at the time of shipment and its default setting when it is initialized should be available in the user guide.

- Legal Disclaimer

  Exclusion clauses that limit the liability of the developers in the event that something goes wrong with an embedded system and the user suffers the damage must be listed.

- Supported Security Standard[22]

  To let the user know that the product is reliable, it is recommended to include the security standards it supports in the manual. Below are some examples of the security standards.

  - Cryptographic algorithms (for example, AES, DES, RC4, RSA)
  - Hash algorithms (for example, SHA-1, SHA-256, MD5)
  - Secure transmission protocols (for example, SSL (TLS), SSH, IPsec)

- Repair Service Information

  The procedure on what the user needs to do when asking for the repair should be provided.

  - Contact information to ask for help and repair
  - Data protection procedures the user should take before sending the product to repair

・ Disposal Procedure

Below should be clearly stated as to what the user should do to remove any sensitive information.

- That the users are solely responsible for removing all data stored in the product.
- How to start the data removal program
- How to use the data removal program (especially how to specify sensitive information)
- How to confirm that the specified data are indeed removed

Level	Description
Level 1	・ There is a user guide but no security is mentioned in it.
Level 2	・ Security issues are covered in the user guide. ・ Trouble shooting and legal disclaimer are included in the user guide. ・ Security requirements (risks and warranty scope) are included in the user guide.
Level 3	・ An organizational procedure to examine whether its security policy and rules are reflected in the user guide is established. ・ Trouble shooting and legal disclaimer are included in the user guide. ・ Security requirements (risks and warranty scope) are included in the user guide.
Level 4	・ An organizational procedure to examine whether its security policy and rules are reflected in the user guide is established. ・ The security guide is regularly updated to include new security risks. ・ Trouble shooting and legal disclaimer are included in the user guide.

---Coffee Break---

Have you ever heard of the “transition to next-generation cryptographic algorithms” or the ”Year 2010 issues on cryptographic algorithms”. Behind this is a concern over the deterioration of security of the standard encryption algorithms that have been widely used around the world since late 1990s. Governments have been calling for safe transition to new, more secure encryption algorithms by around 2010-2013. This “calls” are called the “transition to next-generation cryptographic algorithms” or the “Year 2010 issues on cryptographic algorithms”. In Japan, National Information Security Center has led the effort and published a policy that aims to transit to new algorithms by around 2013.

Deterioration of security of cryptographic algorithms is caused by both the advance of cryptoanalysis techniques and computing power. This means that an algorithm can be safe at some point of time but may become unsafe as time passes. Especially, computing power is rapidly growing each year and has a great impact on the safety of encryption algorithms. Based on the prediction of computing power, such as Moore’s Law, it is possible to assume when an algorithm may become breakable. For example, CRYPTREC (Cryptography Research and Evaluation Committees) assumes that the time that takes to breaks the 1024-bit RSA encryption will become just a year by around 2010-2020 (depending on the computing environment).

Unfortunately, the cryptographic community and the IT vendors do not share the same concern over the security of encryption algorithms. This is because the cryptographic community tends to recommend that we should do something because it is *theoretically possible* that harm may be done when the encryption algorithm is broken but the IT vendors tend not to jump into the discussion on who should cover the cost against the risks that *may theoretically emerge in the future* and settle on sitting back for the time being. Their position is polar opposite, but their logic is understandable. What to do (or not to do) needs to be considered and judged based on the balance between the “possible risk of harm in the future” and the “cost to mitigate the risk at the moment”.

What this suggests is that the “transition to next-generation cryptographic algorithms” should be recognized as one of the higher management issues. We hope that the powers that be will not underestimate the “possible risk of harm in the future”. They should also realize that the biggest concern in the “transition to next-generation cryptographic algorithms” is that it will take time (probably decades) to complete the transition unless someone wield the authority to achieve it and that we will be using the old and new systems with different security levels for a long time till the completion. Especially, it may take 20 or even 30 years, instead of 10, to complete the transition for the embedded systems and this strongly suggests that it is necessary to take into account the “transition to next-generation cryptographic algorithms” from the early design phase.

If you are at the planning phase, select a safe algorithm considering the product and system lifecycle. For example, if the expected system lifecycle is 20 years, you should adopt an encryption algorithm that will be still secure in 2030, such as 128-bit secret key encryption (AES and Camellia), RSA with 2048-bit or longer key or the ellipse curve cryptography with 256-bit or longer key for public key encryption, and SHA-256 for hash function.

4.3.6. Factory Production Control

Embedded system software is usually installed during assembly at the factory. The developers need to make sure that malware does not sneak into the embedded software through the assembly equipments. In addition to the software, it is also necessary to care about the information that identifies the product, such as a sticker printed with the serial number unique to each product and the documents to be enclosed with the product. Depending on how the product is used, these may reveal the user's personal information or financial information. The product with a rewritable capability, such as IC cards, may be installed with personal information after the shipment. This means security at the manufacturing process is also important. For security practices, ISMS (ISO/IEC 27001) provides a good reference. Here, let's pick up a few concrete examples.

- Physical Segmentation

  Depending on the required security levels, segment the space inside the factory with physical barriers and implement access control with IC cards or other means at the doorway where people and goods go in and go out, and monitor their comings and goings with cameras.

- Network Segmentation

  In addition to physical segmentation, it is recommended to segment the network with firewalls and layer 2 switches putting up logical barriers for filtering.

- Log Management of Network and Factory Equipments

  In addition to the access control, it is desirable to collect and manage logs on the networks, production equipments, users and operations in a way that cannot be tampered.

- Commodity Management

  It is necessary to have a system to keep tabs on materials, intermediate products, semi finished products and defective products, and detect if one is smuggled out without permission.

- Intermediate Product Management and Disposal

  If data (including paper documents) in the product check sheets contain sensitive information, be careful how to handle the data. Draw up the procedure in advance in case when the sensitive information must be shared among relevant departments to solve the problem.

- Design Data Management

  Depending on the nature of the products, it is necessary to protect the product's design and other sensitive data from disclosure.

Level	Description
Level 1	No consideration is done.
Level 2	As needed, the following controls and managements, such as the use of checklists, are required at the segments where security-related parts are processed. ・Registry management of operators to control access to production equipments (partitioning). ・Access control of operators to limit the access to the manufacturing area (authentication). ・Operators' labor hour management with strict access control and oversight over what can be bring in and out of the manufacturing area. (labor hour management)
Level 3	
Level 4	

---Coffee Break---

Unlike the office environment, computer equipments at the factory are often running on the special applications or those like Windows Embedded. In some case, security patches cannot be applied to these computers and equipments because the patches may cause a problem with the operation of the special applications, or they have to stick to the old OS that the vendor no longer supports since the special applications do not support the latest OS. Likewise, the use of anti-virus software may be out of option because it may interfere in the operation of the special applications. If part of the computers or equipments is infected with a virus in such environment, infection could spread to all the devices on the factory's local network and it will cost a lot to investigate the cause and recover. Production line may also halted till full recovery. Moreover, depending on the products, the infection of the manufacturing computers and equipments may cause the virus to sneak into the products and harm the user. Until recently, it was mitigated by network-centric countermeasures, such as physically separating the factory network from the organization's intranet or the Internet, putting the firewalls and perform filtering if it is necessary to connect to the Internet, or using a UTM (Unified Threat Management) device with an anti-virus capability.

Lately, however, network is no longer sole likely route of virus infection and the risks through other channels have become higher. Infection via external media, such as USB memory sticks is noteworthy and requires special attention. Often there is no other choice but to use an external media when maintaining the system or installing new data. For the countermeasure, check out virus information[23] and the Report on USB-distributed Malware[24]. In addition, the countermeasures like disabling the auto-run and auto-play feature using the Autorun.inf file or using a removable storage media with an anti-virus capability can be employed. It will be also effective to enforce an operational rule for the use of a USB memory stick that one must check the USB memory stick for viruses on the computer installed with anti-virus software.

It is important to regularly review the security measures and operational rules at the factory accordingly to the computers' and equipments' operational environment and virus trends, and to define what to do when the virus infection is detected.

4.3.7. Support of New Technology

For an embedded system with a network connetivity, it is becoming an critical issue to support IPv6, a new networking technology, and Web technology used as an element of the new age of user interface. As for IPv6, the developers check out the information provided by IPv6 Promotion Council, which works on the validation of interconnectivity and protocols, and IPv6 Technology Council, which works on security and interoperability of IPv6, and consider the following issues.

- Protection of Global IP Address

  Even for a home network, a global address will be assigned to each user's each embedded system instead of per connection. This means the possibility of a direct attack against the embedded system through its IP address that has been protected by NAPT (Network Address Port Translation) in the past will increase.

  Countermeasures:

  - Vulnerability posed by upper applications and middleware are common to both IPv4 and IPv6, thus implement the same level of security measures (such as fortifying the embedded system, implementing vulnerability countermeasures and a method to update firmware when vulnerability is found) and testing.
  - Perform security assessment with tools that support vulnerability in IPv6, such as a vulnerability assessment tool for TCP/IP[6] provided by IPA. The developers must be careful in selecting the assessment tools or analyzing the result since some assessment tools do not support IPv6 nor have enough capability.

- Countermeasure against the Increasing Risk of Attacks Exploiting the Way IPv6 Assigns IP Address

  It is assumed that it will be common for IPv6 to assign a unique IP address. This suggests that once an IP address is found out by an attacker, it may became a target of attacks. If a single IP address is used continuously, the risk of the disclosure of privacy information, such as action history and likes and taste, will increase by analyzing the IP address posted on the bulletin boards or access log of Web servers.

  In some cases, the information unique to a product vendor is included into the IP address when an IPv6 address is configured automatically for a device. An attacker may use this information for reference to exploit vulnerability of a specific device. Moreover, the automatic address configuration of IPv6 will enable the external communication without the user's intention and this can be exploited by an attacker.

  Countermeasures:

  - Especially for embedded systems that require communication via the Internet, use an IP address that does not give away the information about the device or change the address per communication as needed when implementing the automatic address configuration.

- Protection against New Features

  IPv6 newly supports a variety of communication methods using the extension headers and tunneling, such as a standard support of IPsec. On the other hand, this will also give an attacker more options of channels for unauthorized access, and may be exploited to bypass the filtering features such as .firewalls and access

control lists.

Countermeasures:

- Not only the vulnerabilities common to both IPv4 and IPv6 but also those that are unique to IPv6 have been reported[25]. Use every caution when implementing IPv6 and testing it.
- Pay close attention to the latest trend regarding IPv6, for example, an issue on the Type 0 Routing Header that has been removed from RFC from the security aspect.

・ Countermeasures to Protect Web Server (Applications)

When a management interface for the embedded system is implemented using a Web server, there is a possibility that a remote attacker may send maliciously crafted strings to trigger malfunctions.

Countermeasures:

- Harden Web applications (such as applying the latest patch, disabling unused services and ports) and implement vulnerability countermeasures (such as cross-site scripting, cross-site request forgery, injection attack).
- Implement filtering to block unnecessary to use the necessary services and access control using authentication.
- Enable the embedded system to implement anti-virus and anti-tamper measures, or to use the equivalent services available via network.

・ Countermeasures to Protect Web Browser

When an embedded system uses a Web browser to access the Internet, there is a possibility that information assets in the embedded system may be disclosed through the vulnerabilities of the Web browser.

Countermeasures:

- Make sure that the embedded system has the common features to protect Web browsers (such as displaying URLs, checking server certificates, blocking pop-ups and avoiding weak encryption algorithms).
- Make it possible for the user to disable the features that are often exploited by an attacker (such as scripts, viewers and cookies) at the user's discretion.
- Enable the embedded system to implement anti-virus measure and WWW filtering, or to use the equivalent services available via network.

・ Countermeasures to Protect Email Client

When an embedded system uses an email client, there is a possibility that information assets in the embedded system may be disclosed through the vulnerabilities of the email client.

Countermeasures:

- Make sure that the embedded system has the common features to protect the email client (such as anti-virus measure, anti-spam measure, anti-phishing measure and cryptography support), and make it also possible for the user to disable the features that are often exploited by an attacker (such as html display and file attachment) at the user's discretion.

Level	Description
Level 1	• New technology is not taken into account at the design phase and it is not included in the specification documents.
Level 2	• Consideration on new technology at the design phase is up to the project manager and system developers. • No organizational policy is established and an audit is not done or addressed.
Level 3	• An organizational policy on support for new technology that should be followed at the design phase is established. • The design process is preceded following the policy, but the audit is not done.
Level 4	• An organizational policy on support for new technology that should be followed at the design phase is established. • The design process is preceded following the policy and the security department conducts the audit to review it.

4.4. Operation Phase

4.4.1. Handling Security Issues

Security issues of embedded systems are mostly discovered by users and security researchers through the use and study of the product, and reported to the vulnerability handling entities, such as the product developers, distributers and IPA. If the security issues are not handled adequately, it may result in information disclosure or malfunction causing damage to the users, or end up harming the brand image of the product developer or in a lawsuit. Unfortunately, not all vulnerabilities are reported in good faith. Someone may report vulnerability with ulterior motive or it may be used for blackmail or financial gain.

When a security issue (vulnerability) is found in an embedded system, its developer needs to react swiftly to solve the problem. To do that, the developers should set down the following things as an organization.

- Set up a contact point for the users and entities that distribute vulnerability information to accept reports and receive information.
- Draw up the response flow and procedures to prepare a security patch to fix vulnerability.
- Establish the relationship and contact procedure[26] with the relevant organizations (such as JPCERT/CC, other software vendors who may be affected by the found vulnerability).

As for the contact point in particular, the developer should set up a specialized contact point for the purpose and assign someone that could handle the acquired information to swiftly disseminate information to the system developers. To convey the vulnerability information quickly to the system developers, it is advisable to keep tabs on information about the software applications used in the product regardless of whether or not the product is developed in-house, and plan in advance how to make sure that the system developers get the necessary information accurately. Consider to perform a regular exercise to see if the flow and procedure work as planned assuming that a security issue is discovered.

Level	Description
Level 1	・ No effort to respond to security issues (vulnerabilities)
Level 2	・ Vulnerability information obtained through some means is forwarded to those involved in system development. ・ Vulnerability countermeasure is usually implemented by the system developers.
Level 3	・ Vulnerability information obtained through some means is forwarded to those involved in system development. ・ Based on the organization's handling policy, the system developers implement vulnerability countermeasures or decide whether it should be released to the public.
Level 4	・ A specialized contact point and system for vulnerability handling (CSIRT: Computer Security Incident Response Team) and the handling flow are established. In addition, vulnerability information reported from the users or collected from relevant organization is managed by the specialized security team. ・ The time-line and procedure to handle a reported vulnerability is established and the vulnerability information is released to the public through the reporting to IPA or JPCERT/CC.

4.4.2. User Notification and Fixing Vulnerability

If vulnerability of an embedded system is found and a security patch is made and to be distributed, the developer needs to notify the users of the affected product about how to apply the security patch, or how to send it back for recall or repair in some cases. If the affected embedded system is special one and sold to only particular companies, the developer knows to whom it has sold the product and it is possible to make sure that the vulnerability is fixed by notifying the contact person at the client company. On the other hand, if the embedded system is sold in a general market, it is difficult for the developer and distributers to keep tabs on the users and necessary to establish some kind of a user notification system to promote the vulnerability countermeasure to the users as much as possible. If the embedded system has a network capability, it is effective to implement the feature that regularly looks up the information on the security updates. However, it may not be an option since some embedded systems are designed based on the assumption that it does not connect to any network due to the nature of the embedded system. It will be also effective to encourage user registration through the Internet or a registration postcard and provide vulnerability information to those registered.

In the case of digital TV, there is a mechanism to download the security patch through airwave during the time of the day when the most users do not watch TV, such as late in the night or early in the morning. By implementing such mechanism or a network-based update download mechanism into the embedded system, it is possible to support distribution of security patch. There may some cases where the power plug is not connected because of an economical or environmental reason and a security patch cannot be downloaded through airwave. In such cases, it is necessary to call for the users to use it appropriately through the user guide.

When notifying the users, it is necessary to clearly explain that unlike bugs or version ups, leaving vulnerability unfixed will increase the chance of malicious attacks and result in harm for the users. With a little scare tactic, it is hoped that the users acknowledge the difference between a security patch and a version up for additional features and act swiftly to avoid harm. It is necessary to make sure that the update or notification mechanism is not exploited (such as modification of firmware and removal of sensitive information) and the protective measures are implemented (such as server certificate, communication and data encryption, confirmation of data integrity if distributed via the Internet). Below are some examples of the method of user notification.

- By mail
- Through the developer’s Website, TV commercials and announcement on the news paper
- Vulnerability Countermeasure Information Database (JVN)

By reporting vulnerability information to IPA or JPCERT/CC, it is uploaded to the Vulnerability Countermeasure Information Database (JVN iPedia) or the Vulnerability Countermeasure Information Portal (JVN). The Vulnerability Countermeasure Information Database is created to collect and store a multitude of vulnerability information and the Vulnerability Countermeasure Information Portal comes with all kind of efforts to publish vulnerability information in a timely manner. If the developers wonder what kind of information or how much information should be made public, use the Vulnerability Information Disclosure Manual for Software Developer[27] provided by IPA as reference.

Level	Description
Level 1	・ No notification to the users is done.
Level 2	・ A way for the users to actively obtain vulnerability information is provided, such as through the Website of the developer or distributers.
Level 3	・ A direct way for the users to obtain vulnerability information is provided, such as e-mail or letter.
Level 4	・ There is an automatic user notification and a surely-updating mechanism in place..

4.4.3. Leveraging Vulnerability Information

The vulnerability information reported from the users and relevant organizations should be appropriately administered to prevent reemerging or affecting similar products and to leverage the information when the same vulnerability does reoccur. If vulnerability is found all too often, it will damage the corporate brand image and may affect management. To fully leverage vulnerability information, it is necessary to address not only the way to organize a vulnerability response framework and distribute the information and patches but also the root cause of how the vulnerability was created and missed in the first place. Below are some examples of vulnerability of embedded systems reported to IPA in the past.

- Cross-site request forgery vulnerability
- Cross-site scripting vulnerability
- Certificate-related vulnerability
- Vulnerability in the default settings of embedded systems

As for the ways to leverage vulnerability information, the developers could build a database of vulnerability and countermeasure information, establish a work flow to collect and administer vulnerability and countermeasure information and develop an environment where relevant people can browse.

Here are some ways to leverage vulnerability information.

- Turning the information into the upper process to prevent reoccurrence (for example, reviewing test and debug requirements)
- Checking the similar products for the vulnerability

Level	Description
Level 1	・ Vulnerability information is not administered.
Level 2	・ Vulnerability information is administered by the system developers.
Level 3	・ A workflow management system to leverage vulnerability information as an organization is established.
Level 4	・ A workflow management system to leverage vulnerability information and countermeasures as an organization is established and a database for vulnerability related information that is updated through the product development management is built. ・ The database is available for embedded system developers.

---Coffee Break---

Base hardware and software of embedded systems have evolved over the years. For example, the Linux kernel has become major with its version 2.2 released in 1999, and after that, the version 2.4 was released in 2001 and the version 2.6 in 2003. As of 2010, the latest version is 2.6.34 with several minor version-ups since 2003.

The Linux distributions that come with the kernel and various components go through more frequent version-ups. For example, Ubuntu, a popular Linux distribution for client, is upgraded twice a year.

On the other hand, some embedded systems have been in operation more than a decade. Note that the latest software at the time of development may be a legacy that is no longer used nor supported for maintenance.

4.5. Disposal Phase

4.5.1. Promoting User Preparation for Disposal

The data to be disposed are the sensitive information (such as personal information and private contents) that the users stored after acquiring the product. When such sensitive information is stored in the embedded system, it is advisable to enables the users to remove the data with simple operation following the instruction given in the user guide or on the developer's Website. The information that should be provided in the user guide or on the Web site is as follows.

- That the users are solely responsible for removing all data stored in the product.
- How to start the data erasure program
- How to use the data erasure program (especially how to specify sensitive information)
- How to confirm that the specified data are indeed erased

Therefore, it is necessary to implement a standard feature that enables the users to remove the sensitive data when disposing the product. For that, a technical mechanism that identifies the way to identify the sensitive data and securely remove them must be implemented. To be specific, the sensitive data should be stored in a memory space separated from non-sensitive data to identify them and a mechanism to physically remove the data from the storage media need to be implemented. Some user guides and Websites describe a data removal program as "initializing (restoring the factory status)", but it is preferred to make the users acknowledge that they should prepare the product ready for disposal.

For example, in the case of cell phone recycling, the efforts shown below are made to promote removal of personal information before disposal. Below are excerpts from the Policy for Removal of Sensitive Data from Cell Phone by the Telecommunications Carriers Association[28]. To see the specific case studies, check out the information provided by individual telecom carriers[29].

1. There is a feature available that enables the users to remove personal information stored in the cell phone on their own when disposing it.
2. How to remove the personal information is shown in the user manual in an easy-to-follow way or the shop staff will help the customer remove the personal information.
3. There is a service available that takes a backup of the precious data stored in the cell phone to CD-ROM when disposing it. The service may not be available for some models. Please ask the shop staff.
4. If needed, it is also possible to physically destroy the cell phone in front of the customer to make sure that it is not longer usable.

Level	Description
Level 1	・ No consideration on disposal is done.
Level 2	・ Information about disposal and data removal is proved to the users through the user guide and the Website.
Level 3	・ Information about disposal, data removal and recycling is defined as an organization and proved to the users through the user guide and the Website. ・ How to release the information to the users is established as an organization.
Level 4	・ Information about disposal, data removal and recycling is defined as an organization and proved to the users through the user guide and the Website. ・ The products are being tracked and there is a framework to work as an organization when disposing them. ・ Investment to promote the activity is actively made.

1 A threat model developed by the U.S. Microsoft's Security Engineering and Communications that divides the threats into six groups: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

2 Security Technology Map on Threats & Countermeasure of Embedded Software, Research Report (http://www.ipa.go.jp/security/fy18/reports/embedded/index.html)(Japanese)

3 Security in Combined Use of Multiple Embedded Systems, Research Report (http://www.ipa.go.jp/security/fy19/reports/embedded/index.html)(Japanese)

4 JVN Vulnerability Countermeasure Information (http://www.ipa.go.jp/security/vuln/documents/index.html)(Japanese)

5 Purpose of New Edition of Secure Programming Course (http://www.ipa.go.jp/security/awareness/vendor/programmingv2/index.html)(Japanese)

6 TCP/IP Vulnerability Assessment Tool (http://www.ipa.go.jp/security/vuln/vuln_TCPIP_Check.html)(Japanese)

7 SIP Vulnerability Assessment Tool (http://www.ipa.go.jp/security/vuln/vuln_SIP_Check.html)(Japanese)

8 Nessus (http://www.nessus.org/)

9 nmap (http://nmap.org/)

10 Metasploit (http://www.metasploit.com/)

11 nikto2 (http://cirt.net/nikto2)

12 Protos (http://www.ee.oulu.fi/research/ouspg/protos/)

13 JVN iPedia (http://jvndb.jvn.jp/en/)

14 MyJVN (http://jvndb.jvn.jp/apis/myjvn/)(Japanese)

15 Vulnerability Handling (http://www.jpcert.or.jp/vh/)(Japanese)

16 SecurityFocus (http://www.securityfocus.com/)

17 CRYPTREC (http://www.cryptrec.go.jp//english/report.html)

18 A system that keeps tracks on the versions of software and drivers that need to be revisioned to make sure to apply the appropriate version to the products.

19 Purpose of New Edition of Secure Programming Course (http://www.ipa.go.jp/security/awareness/vendor/programmingv2/index.html)(Japanese)

20 Toward Establishment of Methods to Comprehend and Review Software Source Code (http://www.ipa.go.jp/security/fy17/lab/codeblog.html)(Japanese)

21 Injection Detection Tool (http://www.ipa.go.jp/security/vuln/iLogScanner/index.html)(Japanese)

22 It is advisable to pay close attention to the latest trend for the safety of encryption algorithms will deteriorate as time passes.

23 IPA Virus Information (http://www.ipa.go.jp/security/isg/virus.html)(Japanese)

24 JPCERT/CC, Report on USB-distributed Malware (https://www.jpcert.or.jp/research/2009/usbmalware_20090619.pdf)(Japanese)

25 There is a case study where because IPv6's Neighbor Discovery Protocol (NDP) was not properly implemented, communications were intercepted or a denial of service (DoS) condition was created. The information is available in the JPCERT/CC report (http://www.jpcert.or.jp/wr/2008/wr083901.html#1)(Japanese). For more information, JPCERT/CC provides information on Ipv6 specific vulnerability for the system developers.

26 See the IPA Vulnerability Countermeasure Information Handling Guideline (http://www.ipa.go.jp/security/ciadr/partnership_guide.html(2009))(Japanese)

27 See the Vulnerability Information Disclosure Manual for Software Developer (http://www.ipa.go.jp/security/ciadr/vuln_announce_manual.pdf)(Japanese)

28 The Policy for Removal of Sensitive Data from Cell Phone, Telecommunications Carriers Association (http://www.mobile-recycle.net/privacy/index.html)(Japanese)

29 The shops have specialized tools at hand to enforce the physical removal (http://www.nttdocomo.co.jp/corporate/csr/activity/hokkaido/natural_env/)(Japanese)

Appendix

Lifecycle (incl. management)		Level			
Phase	Security-related Items to Consider	Level 1	Level 2	Level 3	Level 4
Management	1-1. Security Rules (P.15)	・No security rules are established.	・Security rules are established voluntarily by the developer. ・Security rules are established by the project manager and system developers.	・Security rules are established as an organization and documented.	・Security rules are established as an organization and documented. An audit to evaluate the compliance to the rules is also in place.
	1-2. Security Education (P.17)	・No security education is done.	・A voluntary working session is held. ・The project manager and system developers acquire the knowledge or organize study sessions as needed.	・Security education is provided as part of daily work. ・Attending security seminars and taking security training are encouraged.	・Security education is provided as part of daily work. ・Taking security training with an expert is required.
	1-3. Collecting Security Information (P.18)	・No security information is collected.	・Security information is collected by the project manager and system developers as needed.	・There exists a system where security information is collected as an organization and disseminated to those involved in system development.	・Is participating in the Vulnerability Information Handling and utilizing vulnerability information.
Planning	2-1. Budgeting (P.19)	・No budget for security issues is ensured.	・Budget is ensured when the project manager requires. ・Budget is considered when the project manager and system developers require.	・There exists a system where a certain amount of budget is allocated to ensure security as part of the development process.	・Budget is allocated to set up a specialized security team and put it on work.
	2-2. Selecting Development Platform (P.21)	・Platform is selected based on cost and delivery period. Security is not considered much.	・Known vulnerabilities at the time of development are taken care of. ・After shipment, nothing is done. ・There is no unit or person in charge that selects the development platform. ・The level of vulnerability awareness differs depending on each project manager and system developer, thus no unified effort as an organization exists.	・When using embedded software packages, monitor information on security patches and update for them released by their vendor. ・Security patches and updates are provided after shipment as well. ・Vulnerability awareness is high in the units that manage the same product and unified effort is being done.	・Vulnerability countermeasures are done across the organization. ・Security patches and updates are provided after shipment as well. ・A vulnerability support system of the vendors is checked out in advance. ・Devices and circuit architecture are selected considering security as well. ・To improve security of products, efforts like building a unique platform or requesting the vendors for improvement are being done.
Development	3-1. Designing (P.26)	・No security is taken into account at the design phase and it is not included in the specification documents.	・Consideration on security at the design phase is up to the project manager and system developers. ・No organizational policy is established and an audit is not done nor addressed.	・An organizational security policy that should be followed at the design phase is established. ・The design process is preceded following the policy. ・An audit is not done.	・An organizational security policy that should be followed at the design phase is established. ・The design process is preceded following the policy and there is an audit process to review the security measures.
	3-2. Software Implementation (P.29)	・No security coding is done. ・No source code review is done. ・Source code reviews are done but security is out of scope.	・Coding policy and review technique to prevent vulnerability depend on the voluntary effort by system developers or some project members. ・No uniformed effort is done even in the same department. ・Because it is done based on the knowledge and experiences of the project manager and system developers, there are no clear criteria.	・There is an organizational policy and rules about coding and reviews that should be followed in software implementation to prevent vulnerability. ・System developers write source code following the rules. ・An audit of the compliance with the rules within or across the departments is included in the development process.	・In addition to the efforts at the level 3, there is a specialized security team in the organization and it gives supports for secure cording and reviews. ・The security team is also involved with the rule making and keeps revising the rules based on the information available on bugs, vulnerabilities, incidents and countermeasures
	3-3. Outsourcing (P.30)	・No vulnerability countermeasure is taken into account when outsourcing. ・Even if the project has security features, the same outsourcing policy is applied.	・Vulnerability countermeasure in the case of outsourcing is up to the voluntary effort by system developers or some project members. ・No uniformed effort is done even in the same department. ・Because it is done based on the knowledge and experiences of the project manager and system developers, there are no clear criteria.	・A contract for software development outsourcing requires an organizational security effort within the outsourced party to prevent vulnerability. ・An audit of the compliance with the contract conditions within or across the departments is included in the development process.	・In addition to the efforts at the level 3, there is a security team in the organization and it gives advice to the outsourced entity.
	3-4. Security Assessment Test and Debugging (P.33)	・No security testing is done.	・Security test is planned based on the knowledge and experience of system developers. ・The quality of the security testing depends on system developers' knowledge.	・There is an organizational security test plan and evaluation is done based on the organization's criteria. ・If other device is connected, test the connectivity with the device.	・Following the security the test plan drawn up as an organization, the internal security team reviews the result. ・If something is added, updated or delete after shipment, check the connectivity with the new device.
	3-5. User Guide (P.35)	・There is a user guide but no security is mentioned in it.	・Security issues are covered in the user guide. ・Trouble shooting and legal disclaimer are included in the user guide. ・Security requirements (risks and warranty scope) are included in the user guide.	・An organizational procedure to examine whether its security policy and rules are reflected in the user guide is established. ・Trouble shooting and legal disclaimer are included in the user guide. ・Security requirements (risks and warranty scope) are included in the user guide.	・An organizational procedure to examine whether its security policy and rules are reflected in the user guide is established. ・The security guide is regularly updated to include new security risks. ・Trouble shooting and legal disclaimer are included in the user guide.
	3-6. Factory Production Control (P.37)	No consideration is done.	As needed, the following controls and managements, such as the use of checklists, are required at the segments where security-related parts are processed ・Registry management of operators to control access to production equipments (partitioning). ・Access control of operators to limit the access to the manufacturing area (authentication). ・Operators' labor hour management with strict access control and oversight over what can be bring in and out of the manufacturing area. (labor hour management)		
	3-7. Support of New Technologies (P.41)	・New technology is not taken into account at the design phase and it is not included in the specification documents.	・Consideration on new technology at the design phase is up to the project manager and system developers ・No organizational policy is established and an audit is not done or addressed.	・An organizational policy on support for new technology that should be followed at the design phase is established. ・The design process is preceded following the policy, but the audit is not done.	・An organizational policy on support for new technology that should be followed at the design phase is established. ・The design process is preceded following the policy and the security department conducts the audit to review it.
Operation	4-1. Handling Security Issues (P.42)	・No effort to respond to security issues (vulnerabilities).	・Vulnerability information obtained through some means is forwarded to those involve in system development. ・Vulnerability countermeasure is usually implemented by the system developers.	・Vulnerability information obtained through some means is forwarded to those involved in system development. ・Based on the organization's handling policy, the system developers implement vulnerability countermeasures or decide whether it should be released to the public.	・A specialized contact point and system for vulnerability handling (CSIRT: Computer Security Incident Response Team) and the handling flow are established. In addition, vulnerability information reported from the users or collected from relevant organization is managed by the specialized security team. ・The time-line and procedure to handle a reported vulnerability is established and the vulnerability information is released to the public through the reporting to IPA or JPCERT/CC.
	4-2. User Notification and Fixing Vulnerability (P.43)	・No notification to the users is done.	・A way for the users to actively obtain vulnerability information is provided, such as through the Website of the developer or distributers.	・A direct way for the users to obtain vulnerability information is provided, such as e-mail or letter.	・There is an automatic user notification and a surely-updating mechanism in place.
	4-3. Leveraging Vulnerability Information (P.45)	・Vulnerability information is not administered.	・Vulnerability information is administered by the system developers	・A workflow management system to leverage vulnerability information as an organization is established.	・A workflow management system to leverage vulnerability information and countermeasures as an organization is established and a database for vulnerability related information that is updated through the product development management is built. ・The database is available for embedded system developers.
Disposal	5-1. Promoting User Preparation for Disposal (P.46)	・No consideration on disposal is done.	・Information about disposal and data removal is proved to the users through the user guide and the Website.	・Information about disposal, data removal and recycling is defined as an organization and proved to the users through the user guide and the Website. ・How to release the information to the users is established as an organization.	・Information about disposal, data removal and recycling is defined as an organization and proved to the users through the user guide and the Website. ・The products are being tracked and there is a framework to work as an organization when disposing them. ・Investment to promote the activity is actively made.

This page intentionally left blank.

For more information:

IT Security Center, Information-technology Promotion Agency of Japan

Bunkyo Green Court Center Office, 16th Floor,

2-28-8 Hon-Komagome, Bunkyo-ku, Tokyo, 113-6591, Japan

TEL: +81-3-5678-7527　FAX: +81-3-5978-7518

E-mail：vuln-inq@ipa.go.jp　URL：http://www.ipa.go.jp/security/english/

This guide is available for download at:

http://www.ipa.go.jp/security/fy22/reports/emb_app2010/emb_guide_fy22_eng.pdf

2009/6/24	First Edition
2010/9/7	Revised Edition
2011/2/22	English Edition