Common Industrial Control System Vulnerability Disclosure Framework

08 - June -2012

Contents

1. Executive Summary

The Industrial Control System Common Vulnerability Disclosure Framework ('the Framework') is intended to provide consensus-based guidance to vendors and systems integrators for the creation of Vulnerability Disclosure Policies. The Industrial Control System (ICS) industry has faced strong criticism in past years for poor disclosure of potential vulnerabilities in critical infrastructure (CI) products. Desire to demonstrate a strong commitment to security and to positive handling of vulnerabilities has led to a focus on responsible disclosure; however vendors are challenged by a lack of clear industry-specific guidance. Inconsistent disclosure policies have also contributed to a public perception of disorganization within the ICS security community.

2. Document Purpose

This document is intended to provide a consensus-based foundation for ICS vendors and integrators working to develop a vulnerability disclosure policy. This document is not intended to be prescriptive. Recognizing that each vendor has unique challenges relating to how their products are used and the potential risks inherent in different disclosure paths, this document provides recommended ranges and formats for different aspects of the disclosure process. The information contained in this document should be taken as building blocks towards the creation of a policy

3. Document Expectations

This is a living document and will continue to evolve to reflect the expectations of both asset owners and the IT community in general. As well, new government regulations and best practices will impose additional requirements on disclosure policies.

4. Software Vulnerabilities

Most asset owners associate vulnerability disclosure specifically with software vulnerabilities. This super-class of vulnerabilities encompasses a number of different categories of software issues, each of which must be evaluated independently.

It should be noted that the software vulnerabilities discussed below are first-order vulnerabilities and do not include issues relating to software vulnerabilities identified in third-party products. While the categorization of the potential issues is the same, the opportunities available for, and the impact of mitigations can be quite different.

4.1 Types of Vulnerabilities and Associated Remediation

4.1.1 Architectural Vulnerabilities

Architectural vulnerabilities in software occur when the design of an application or component exposes a security flaw despite the use of appropriate coding standards. Architectural flaws typically occur in the application design phase and can result from insufficient threat modeling within the secure development lifecycle.

Architectural vulnerabilities may also be created through the maintenance of legacy support. Providing access to both legacy interfaces as well as more secure interfaces allows an attacker to select the weaker of the access methods when attempting to subvert the application. Legacy support decisions may also result in APIs being expanded to support secure connections from new client applications without deprecating and removing methods that bypass a strong security model.

The deployment of the ICS application may also result in the exposure of architectural vulnerabilities. Many ICS solutions are designed to be deployed in a particular configuration and rely on additional protections being available in order to function securely. An example of this would be a solution that requires a legacy network protocol to communicate with a particular hardware component requiring a secure network to mitigate potential risks. If this solution is deployed onto an insecure network without the appropriate protections in place, it could result in the exposure of the vulnerable protocol. The

vulnerability is architectural in nature as it is a weakness in the design of the solution, once the environment into which it is deployed is accounted for.

Examples of architectural flaws include:

- Internal services exposed to external connections without appropriate security
- Communication channels existing between multiple hosts without adequate authentication or authorization elements
- Inadequate or absent input parsing on server or middleware applications allowing injection attacks
- A lack of cryptographic protection for business sensitive communications

4.1.2 Remediating Architectural Vulnerabilities

Architectural vulnerabilities can be among the most challenging to appropriately mitigate. In many situations it is necessary to sacrifice functionality or to significantly alter the configuration of an existing solution to resolve this class of issue. This may result in an extended remediation timeline, requiring the re-release of one or more application products. Often, architectural issues are too complex to be resolved through application hotfixes, though this is not always the case.

Mitigation of architectural vulnerabilities while remediation is pending may include:

- Application of third-party security solutions
- Use of hardware controls to reduce exposure
- Detection of attacks or exploits through software or hardware
- Disabling of product functionality to prevent access to vulnerable components

4.1.3 Implementation Vulnerabilities

Implementation (code-based) vulnerabilities exist when the implementation of a feature or function within an application is done without applying secure coding principles and practices. These vulnerabilities are the generally the most visible to asset owners as they are regularly identified and patched through the normal product lifecycle.

Application crashes or failures when exposed to particular types of input or use cases may be indicative of implementation vulnerability however this cannot be considered conclusive. Further, these behaviors do not provide an indication of the scope of a given vulnerability if it does exist.

Internal product reviews are the most effective mechanism for the detection of implementation vulnerabilities. With access to the source code for an application, a wide range of methods can be leveraged to recognize potential security risks. These can include;

- Manual code reviews
- Static code analysis
- Compile-time functional analysis (scanning for deprecated / insecure functions)

Tools commonly associated with external analysis are appropriate for internal testing as well. In some cases, use of these tools with access to source code and design knowledge may improve testing effectiveness for internal investigators. In particular, fuzz testing is a useful method to exercise parsers, input validation, and exception handling. Unexpected observations may indicate code implementation vulnerability or other input trust issues.

Tools which can be used to identify programmatic vulnerabilities without access to the source code of the solution include;

- Fuzzing suites
- Port / service mapping tools
- Vulnerability scanning tools
- Vulnerability exploitation frameworks
- Binary reverse-engineering tools

With external analysis tools, it is often difficult to determine the specific flaw within the application. Many vulnerabilities identified by third parties are found through the use of these tools however access to the application source is required to successfully locate the root cause of the problem. For this reason third parties who identify issues with products cannot typically provide remediation. The application vendor must take the information generated by the external test tools and use it as the foundation for an investigation.

4.1.4 Remediating Programmatic Vulnerabilities

In many cases programmatic vulnerabilities can be remediated through relatively minor changes to the application code. These types of corrective measures lend themselves to ‘patch’ or ‘hotfix’ release, in which particular components of the application are replaced with newer components that no longer contain the vulnerable code. Note that this is not possible in all cases.

Programmatic vulnerabilities do not typically require a full release of a product in order to achieve remediation.

4.1.5 Third-Party Software Vulnerabilities

Third-party vulnerabilities affect software components used in the first-party/vendor's products. Like first-party vulnerabilities, third-party vulnerabilities can be design or implementation. Contemporary software commonly makes extensive use of third-party application code, creating the potential for vulnerabilities which affect the product but are not within the direct control of the software vendor. Third-party products used can range from individual libraries to embedded applications, and may also include independent applications which are sold and used along with the vendor’s own proprietary solution.

It is often challenging for a vendor to resolve a third-party vulnerability as they typically have no direct control over the application. Testing requirements are also significantly greater as the nature of the changes to the third-party product is typically unknown. The vendor must re-test all aspects of their application which interact with the third-party component to verify that the new version, whether updated or replaced, remains compatible with product expectations.

4.1.6 Remediating Third-Party Vulnerabilities

In the case of independent but bundled applications, provided that the third-party vendor does not dramatically alter the core functionality of the product, security updates and patches can be applied with limited risk.

Similarly, embedded applications may be upgradeable without major impact to the ICS solution depending on the type of software. An embedded calculation engine may be high risk to upgrade, while a third-party screen-shot utility may be a straightforward upgrade.

Patches or updates that impact third-party libraries or toolkits are often very complex to update within production environments. Extensive testing is usually required to validate that all critical functionality remains stable after a change. In many cases, these types of applications are not updated without a major product release so that full functional testing can take place.

Asset owners typically have a lower level of visibility into the software which is embedded in a product. If the vendor judges the application of a patch to a third-party product to place the integrity or stability of the product at risk, the product should be tested and the patch validated by the vendor before being applied.

It is important to note that while the application vendor may not have direct control over the third-party applications embedded in their products, they retain an obligation to exercise due diligence in ensuring the safety and security of their end customer. To this end, if the third-party component vendor does not demonstrate an appropriate responsiveness and begin a remediation and disclosure process which meets the standards set out in this document, the application vendor must attempt to do so within the bounds of their legal agreements with the third-party supplier. In situations where it is

possible to replace or otherwise alter a third-party component which has a vulnerability which will not be mitigated, the vendor must investigate this as a possible remediation path. In the case of products not in active development, or for which there is no reasonable mechanism to resolve the identified issue within the product, the vendor must then make all reasonable attempts to provide a mitigation option to asset owners and to communicate the nature of the risks associated with the specific vulnerability to their customer-base. The overriding expectation in this situation is that the primary vendor act as a shepherd for this applications embedded in their product and where the behavior of the third-party vendor fails to meet the requirements of the asset owner and the critical infrastructure community, the original vendor will make all reasonable efforts to assure the safety and security of their products.

4.2 Mechanisms for Identifying Vulnerabilities

Security vulnerabilities, particularly in the context of a disclosure policy, are typically seen as something that is presented to a company. Whether coming from a third-party, via a mailing list, or the front page of a national newspaper, this 'reactive' perception is alarmingly common. The most common discovery scenarios are much more mundane and therefore present more complex disclosure challenges. When someone outside of your company is aware of a software vulnerability, disclosure is a matter of management. When the vulnerability has been detected via another of the mechanisms discussed below, there are more options in handling disclosure and greater challenges in ensuring responsible handling.

In defining a vulnerability disclosure policy, care must be taken to determine how each of the following scenarios will be handled so that all parties involved have similar expectations. In particular, thresholds should be set internally as to which types of discovery warrant public disclosure, or the level of public disclosure warranted.

4.2.1 Internal Vulnerability Discovery – Design / Development Time

It is not uncommon for security issues to be identified during the design and development of a product. Whether during the creation of a new product or the ongoing evolution of an existing code base, developers should be routinely applying appropriate security analysis. This may include threat modeling, security-centric code reviews, or automated security testing of the product. Any of these can identify the presence of security vulnerability.

If the potential vulnerability identified resides within newly implemented application code which has not yet been publically released, it is typically not treated as a vulnerability requiring disclosure. It is mitigated internally and may be processed through an internal analysis model to determine how the issue may be avoided in the future. This applies to both new applications and new code for pre-existing applications.

4.2.2 Internal Vulnerability Discovery – After Release

Software development groups typically maintain ongoing security assessments of their products, working to identify potential issues ahead of attackers or other external groups. These efforts, in combination software security measures used in the development of enhancements to existing products, create many opportunities to detect potential vulnerabilities internally. When vulnerabilities are detected in this manner, several decisions need to be made in determining how to respond.

Depending on the severity of the vulnerability it may be necessary to notify customers immediately, in advance of the development of a software patch or hotfix. Particularly where mitigations are available which could reduce or eliminate the risk that the vulnerability poses to customers, early notification can be critical to protecting installations.

For vulnerabilities which are considered to be of very low risk and which can be mitigated through a quickly deployed hotfix or patch, it may not be necessary to provide notification until the solution is available. Similarly for vulnerabilities that cannot be effectively mitigated it may be appropriate to delay widespread disclosure until a solution is available. This is a more significant consideration in

situations where the disclosure of the vulnerability may place customers at risk without appropriate mitigations being available.

For exceptionally high risk vulnerabilities which expose customers to significant threats, disclosure of an internally discovered vulnerability is highly recommended even in situations where a resolution is not available.

4.2.3 External Vulnerability Discovery – Customer Discovery

Several factors must be evaluated when a customer identifies a potential vulnerability within a product. Depending on the nature of the security vulnerability the customer may be interested in an immediate resolution or at minimum a timeline for a solution. As well, there is often reluctance on the part of the customer responsible for the discovery of a vulnerability to disclose the potential issue as it may be perceived as placing their installation at greater risk.

Various legal documents may govern the relationship between a vendor and a customer. The nature of the specific framework in place will vary between relationships however it may not explicitly define the handling of information relating to vulnerabilities. An example of this type of governance would be the presence of a non-disclosure agreement between both parties. Care must be taken to ensure that all in-place agreements are observed throughout the disclosure process.

Much like internally-discovered vulnerabilities, customer-discovered vulnerabilities may benefit from selective disclosure. While customers may be very resistant to publically disclosing a vulnerability which impacts their installation they may be more amenable to sharing the information with other customers via a managed and access-controlled mechanism.

The customer-discovery scenarios discussed here are based on the assumption that the customer will contact the vendor through a private channel to make them aware of the vulnerability. In cases where this does not occur and the customer discloses the information to the public, the 0-Day disclosure model applies.

4.2.4 External Vulnerability Discovery – Customer Audit Discovery

While most customers do not actively attempt to validate the security of the products they have purchased at a level likely to identify security vulnerabilities, many do choose to employ third-party security companies to audit the installations and the software. These audits may take place without the knowledge of the vendor however once results have been generated many customers choose to share the resulting report.

In many audits the issues identified relate to the configuration of the product, problems with the underlying operating system configuration, or errors in the network design or configuration. This is not typically due to a lack of potential issues with the products deployed so much as it is a result of the limited depth of the audit. From a customer perspective there is often very limited return on investment for an in-depth audit of any software product. Most often product security vulnerabilities discovered during audits are the result of another type of automated testing, rather than a deliberate attempt to find software issues.

Under standard audit contracts, the results of the audit are confidential to the organization customer and any party that they choose to share those results with. This allows for information to be passed back to the vendor without violating the terms of the audit. The standard contract will also prevent the auditing company from being able to disclose any findings publically. It is important to note however, that it is not required for a customer to pass audit results on to a vendor unless explicitly noted in their contract or software license agreement. As well, the customer may consider the information in the audit report to be confidential to their organization and therefore protected under the existing Non-Disclosure Agreement between the customer and the vendor. In those cases the customer must be in agreement with the planned disclosure and appropriate legal documentation should be prepared detailing the scope of any information disclosure. Even though the vulnerability exists within the vendor's application, they are not necessarily free to disclose it as they see fit. If any doubt exists as

to whether or not an appropriate framework exists to support disclosure of information found during a customer audit a legal review should take place.

It should also be noted that many software license agreements, if not all, prohibit reverse engineering a product. They may also preclude other types of invasive testing aimed at identifying security issues.

4.2.5 External Vulnerability Discovery – Independent Researcher

The term 'independent researcher' can be used to describe a single individual or an organized company. Independent researchers actively assess products to identify security issues on their own time and funding. Depending on the product involved, they may or may not have legitimate access to the product in question. This is a key factor in that if an individual or group has not legally obtained access to the product they are not bound by the software license agreement and may use techniques which are proscribed. Independent researchers are also not bound by NDA with either the vendor or a customer, giving them a high level of flexibility as to how and when they choose to communicate any vulnerability they find.

Independent researchers, like auditors, may not have detailed information on the vulnerability they have found. Even in cases where the cause of the vulnerability could be identified it is often time consuming. Vulnerabilities identified by independent researchers may contain limited information, lacking details required to reproduce the issue quickly or potentially containing incorrect information. This places an increased burden on the vendor to rapidly determine the nature of the vulnerability and to evaluate the risk.

Independent researchers are the first discovery mechanism that does not have an established path through which to provide information to a vendor. For Internal, Customer, and Customer-Audit discoveries, a communication channel exists as well as a legal framework to govern interactions. Independent researchers typically lack access to any of these and must therefore build a communication path to the vendor in order to make them aware of a potential vulnerability. If this task is overly onerous or does not provide the level of feedback the researcher is expecting, they may choose to pursue an independent disclosure. In this scenario the researcher does not have a direct stake in keeping any potential vulnerability confidential and therefore the risk of their disclosing information on their own is significantly higher.

Mechanisms to facilitate communication with independent security researchers may include easy to identify security or support email addresses, as well as information submission forms available through descriptively named pages on a company website;

Examples:

security@company.com

www.company.com/security

Product Support channels may also be an appropriate path for information submission (ex. info@company.com) however a process must be in place to capture security-related information and send it to the correct team internally rather than having it queued with other support requests. It is important to note that bug or vulnerability submission forms that are only available to customers through a protected Extranet or similar tool are not available to independent researchers.

Failure to provide an appropriate channel through which a researcher can present information to your company can encourage a '0-day' situation.

Independent researchers may also notify vendors through a third-party, often a Computer Emergency Response Team (CERT) or Computer Security Incident Response Team (CSIRT) that provides vulnerability coordination services. A coordinator typically acts as a broker between the independent researcher and vendor to improve communication and help manage the disclosure process. Like an independent researcher, a coordinator may have their own disclosure policy that will need to be considered. The Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) provides coordination services for ICS vulnerabilities. A coordinator can also assist vendors responding to vulnerabilities.

4.2.6 External Vulnerability Discovery – 0-Day

A '0-day' disclosure occurs when a vulnerability is released prior to the vendor being made aware. '0-day' vulnerability disclosures do not have fixes immediately available and are often publically disclosed through mailing lists or forums. In these situations, both the customers and the vendor are at a disadvantage as they try to validate the existence of the potential vulnerability, identify mitigations, and develop a long-term fix if appropriate while malicious parties may have access to the exploit.

5. Types of Disclosure

The following section discusses three different disclosure scopes. These are not firm definitions but should serve as starting points for developing a scope which best suits your company's business and customers. Not disclosing an issue is not discussed; however it remains an option and may be appropriate in some scenarios.

5.1 Private Customer Disclosure

Provided that a vendor is able to contact all customers who are using a potentially vulnerable product disclosure can be restricted to only the affected install base. This disclosure model can be particularly advantageous when only a small number of customers are affected by a vulnerability as it reduces the risk of information becoming available to malicious parties while ensuring that all affected users are advised of appropriate mitigations. After a hotfix has been provided to customers and the threat has been eliminated or greatly reduced, a broader disclosure may be appropriate.

Note that the success of this type of disclosure depends on being able to reliably contact all impacted customers. This type of disclosure is also more successful if the information can be transmitted to customers securely, and the extent to which customers can maintain confidentiality of the information.

5.2 Public Disclosure

Public Disclosure entails notifying the broader IT community through a controlled mechanism such as a website post, blog entry, or security advisory. Public Disclosure does increase risk to customers, as any information disclosed about the vulnerability is available to malicious individuals as well as to legitimate customers. If a vulnerability is disclosed publically prior to a fix being made available, or prior to an available fix being deployed to all customers, malicious parties may be able to use that information to impact customer operations.

The availability of a fix does not eliminate the risk inherent in a public disclosure. For customers who have not installed the available fix, public disclosure with an accompanying publically available hotfix may increase risk. When a hotfix is available, malicious parties may be able to reverse engineer the hotfix code and use that to develop an exploit faster than they might otherwise be able to.

Public Disclosure must be carefully considered to ensure that all risks are appropriately mitigated. A balance must be found between the increased risk of broader disclosure and the challenges of reaching all impacted customers in a reasonable time frame.

5.3 Third-Party Disclosure

Third-party disclosure typically involves an independent group disclosing vulnerability without the consent of the vendor. However, this is not always the case. The disclosing party may include a vendor-neutral entity such as ICS-CERT working on behalf of a vendor to communicate information. In most cases where disclosure is fully planned the vendor takes a primary role in releasing information about the vulnerability. In some cases however, a researcher may have entrusted a third-party group to manage a vulnerability and to coordinate its resolution with the vendor. If the vendor has failed to meet specific objectives associated with the standing agreement, the third-party may be empowered to release the information on a specific date.

Third-party disclosure scenarios usually provide a level of warning to the vendor, allowing them to prepare for the customer and community response to the vulnerability. They do not offer the same

level of control that the vendor would have were they to handle the information release on their own, however.

6. Vulnerability Disclosure Policy Components

The following components will form the basis for a formal Vulnerability Disclosure policy. When these have been defined for your organization you should be able to define an appropriate course of action for each of the different scenarios listed previously.

6.1 Purpose of the Vulnerability Disclosure Policy

Each company must clearly understand the purpose of their vulnerability disclosure policy and must then state this in the policy itself. This component is a key to ensuring that customers as well as employees understand the intent of the policy and that the decisions they make are focused on the spirit of the policy. It is impossible to cover all eventualities however a clear goal will be invaluable in making situational decisions.

It is important that even often-unspoken objectives be highlighted in this statement. Noting that one of the purposes of the policy is to ensure that due diligence is performed by the vendor to minimize legal liability is appropriate. Stating that the policy is intended to empower customers to take active responsibility for implementing recommended mitigations and patching systems quickly can be another mechanism to clearly set out roles and expectations.

6.2 Policy Commitments

The core purpose of the policy itself is to define specific commitments by the vendor to other parties regarding the handling of product security vulnerabilities. It should be kept in mind that this is explicitly not limited to customers. The vulnerability disclosure policy is a public covenant which must apply equally to customers, researchers, and other groups to clearly set expectations for the vendor's behavior.

The commitments made in this section should not be considered constraints to be adhered to even in the face of a path which better achieves the declared purpose of the document. Instead, this should be taken as an intended path with recognition that if changes are required to reflect unique situations they will be communicated to the involved stakeholders.

6.2.1 Policy Deliverables

Through the course of responding to a product vulnerability various deliverables will be created. These may include hotfixes, mitigation documents, resolution timelines, or impact statements. Each of these deliverables should be defined in the Vulnerability Disclosure Policy so that customers and researchers clearly understand what to expect at each phase in the remediation process.

6.2.2 Timelines

For each part of the disclosure process specified, a timeline should be stated. This will provide customers and researchers with appropriate expectations not only for the delivery of a solution, but also for the provision of mitigations and progress updates. Care should be taken to state that timelines are best-effort, however there should be regular checkpoints established at which the vendor will send out an update on progress and any timeline changes. By establishing checkpoints in advance, other parties will not feel that the vendor is being non-communicative.

In situations where wider disclosure may be delayed, groups such as ICS-CERT may be used to validate timelines. Particularly in situations where a neutral organization has been used in the vulnerability identification process, they can serve as a reference to support that the disclosure policy was followed even when the process is not publically visible.

6.2.3 Mitigations

The Vulnerability Disclosure policy should explicitly specify the process through which mitigations will be communicated to customers as well as the circumstances under which mitigations will be provided. If appropriate, the support that will be provided for the implementation of mitigations can be detailed.

6.2.4 Resolution

The Vulnerability Disclosure policy should clearly state the conditions required for a particular vulnerability disclosure process to be closed. Resolution is most commonly through the closure of the potential vulnerability via a hotfix or patch. However this is not always possible. In some situations, particularly involving architectural vulnerabilities, the issue may be closed from the perspective of the Vulnerability Disclosure process when mitigations are provided and a future release eliminating the vulnerability is defined.

If resolution thresholds are not clearly defined it may be difficult to demonstrate that the disclosure process has been completed, increasing potential for dissatisfaction. When a vulnerability has been submitted by a customer or a third-party researcher this may discourage future submissions.

6.3 External Deliverables

The following deliverables should be externally published.

6.3.1 Vulnerability Disclosure Policy Statement

The Vulnerability Disclosure Policy should be posted publically. The success of the policy in controlling expectations and establishing a predictable response to security vulnerabilities depends on its being accessible to all parties involved in the vulnerability discovery process.

6.4 Contact Mechanisms

Multiple mechanisms should be available for customers and third-party groups to submit a newly discovered vulnerability to a vendor. These mechanisms should be straightforward and be directed to the team responsible for evaluating potential security risks. If at all possible, common support contact information and help-desk systems should not be used unless a strong internal process is in place to ensure that vulnerabilities do not become lost in normal product support loads.

When putting contact mechanisms in place it is important to ensure that it is widely accessible. Information on submitting a vulnerability should not be located behind a password-protected portal or inside a customer forum. If possible the contact mechanism should be exposed to search engines to make it even easier to locate.

6.4.1 Security Webpage

Mechanisms for contacting the vendor's security team, including vulnerability submission, should be presented in a single web page. In addition to containing contact information or submission forms this page can also serve as a means to publish security information when broad public disclosure is appropriate.

This page should be linked from either the main vendor web page, or from the product or vertical page in larger organizations. Where possible, the page should have an easily identified URL such as:

www.vendor.com/product_vertical/security

or

www.vendor.com/security

or

www.vendor.com/support/security

6.4.2 Security Email Address

Similar to the security-focused webpage, the vendor should provide an email address that can be used to submit newly discovered vulnerabilities. This address should be posted in an easy to locate spot and should be connected to a regularly-monitored, shared in-box. In order to ensure the rapid response required for successful vulnerability management, this email address should be monitored daily.

If possible the security email address should be easy to remember and should be logically named.

Example:

security@vendor.com

or

vulnerability@vendor.com

6.4.3 Anonymous Submission Form

An anonymous vulnerability submission form should be available, recognizing that email generally requires that the sender expose their identity. This form should be as complete as possible, allowing for the attachment of code samples and the provision of large amounts of information. Given that the submitter may not be reachable after their submission; all key information needs to be gathered in the single form submission. In designing this form, Product Support or Product Security groups should be consulted to determine what information they require to properly evaluate or review a potential vulnerability.

Note that the individuals using this type of form may have a high level of technical skill and will likely be able to identify any 'tracking mechanisms' embedded in the page or form. Care should be taken to ensure that the anonymity of the form is not only stated, but is also demonstrated.

6.5 Classification of Vulnerabilities

Severity classification is often a significant factor in determining how a vulnerability will be handled under a disclosure policy. If vulnerability severity is used to define specific disclosure paths that will be followed, it is critical that severity levels are defined explicitly within the Vulnerability Disclosure policy. Failure to define the severities levels in terms of both potential impact and

A number of common vulnerability classification models exist within the IT / Security sectors that can be used either as a starting point, or adopted as a classification scheme. [D1][D3][D4]

A widely accepted standard for vulnerability identification is Common Vulnerabilities and Exposures (CVE). When publicly disclosing a vulnerability, a vendor can obtain CVE ID numbers from a CVE candidate numbering authority. [D2]

A CVE ID does not take the place of a vendor's own identification or tracking scheme.

Using CVE helps customers, vulnerability databases, and other stakeholders correctly identify and disambiguate different vulnerability reports.

One method to estimate the severity of a vulnerability report is the Common Vulnerability Scoring System (CVSS). While CVSS cannot account for many of the contextual factors that influence prioritization of the response to a vulnerability, CVSS does provide a reasonable first-order approximation of severity based on the inherent characteristics of the vulnerability.

Appendix A – Terminology

Vulnerability

"In computer security, a vulnerability is a weakness which allows an attacker to reduce a system's information assurance."

Cited: March 28, 2012

URL: http://en.wikipedia.org/wiki/Vulnerability_(computing)

Vendor

"A vendor, or a supplier, is a supply chain management term meaning anyone who provides goods or services to a company."

Cited: March 28, 2012

URL: http://en.wikipedia.org/wiki/Vendor_(supply_chain)

0-day

"A zero-day (or zero-hour or day zero) attack or threat is a computer threat that tries to exploit computer application vulnerabilities that are unknown to others or the software developer. Zero-day exploits (actual software that uses a security hole to carry out an attack) are used or shared by attackers before the developer of the target software knows about the vulnerability."

Cited: March 28, 2012

URL: http://en.wikipedia.org/wiki/Zero-day_attack

Coordinator

"a position within an organization or business with significant responsibilities for acting as a liaison between departments, stakeholders and information sources, which requires many non-administrative competencies."

Cited: March 28, 2012

URL: http://en.wikipedia.org/wiki/Coordinator

Exploit

"An exploit (from the verb to exploit, in the meaning of using something to one's own advantage) is a piece of software, a chunk of data, or sequence of commands that takes advantage of a bug, glitch or vulnerability in order to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic (usually computerised)."

Cited: March 28, 2012

URL: http://en.wikipedia.org/wiki/Exploit_(computer_security)

Appendix B – Sample Disclosure Policy Overview

The sample disclosure policy is intended as a consensus-based foundation for developing a vulnerability disclosure policy. Vendors and integrators are encouraged to adopt similar policy elements to facilitate a more organized industry wide approach within the ICS security community.

B. Sample Disclosure Policy – Key Elements

Key elements in the sample disclosure policy include contact information for reporting a vulnerability, eligibility clarification, discoverer interaction, disclosure entitlements, and handling scenarios.

B1 Contact Information

- Operating hours and supported communication methods email, phone, web, etc…
- Communication response time norms, where applicable, provide instructions for emergency response
- Special instructions regarding transmission of sensitive details

B2 Eligibility Clarification

- Encourage customers to report vulnerabilities regardless of active service contract or product lifecycle status.
- Welcome vulnerability reports from researchers, industry groups, partners, and any other source.
- Agree to triage any vulnerability reasonably believed to be related to vendor's products or services.

B3 Discoverer Interaction

- Collaborate with the discoverer to understand and reproduce the vulnerability.
- Maintain regular communication with the discoverer on case status throughout the disclosure process until reaching a clearly defined point of resolution.
- Ensure the discoverer understands the vendor's position on vulnerability disclosure.
- With the discoverer's consent, provide credit for proper reporting and collaboration.

B4 Disclosure Entitlements

- Issue summary with vendor tracking number and CVE reference (public disclosure)[D2]
- Identity of known affected products and versions
- Information on mitigating factors and workarounds
- Timeline and entitlement for availability security updates

B5 Handling Scenarios

Handling scenarios are useful to outline norms regarding disclosure preferences and thresholds.[D5]

ICS-CERT is the preferred CERT for coordinating a public disclosure. If a vendor prefers a private disclosure method, the policy should describe the method(s) used to reach affected customers.

B5.1 Prioritization Method

The vendor prioritizes vulnerability disclosure handling primarily based on severity. Priority of affected component(s) in a system of systems is a secondary factor. The methods used to classify vulnerability severity and component priority are documented. [D1][D3][D4][D6]

Overriding factors affecting overall handling priority include known active exploitation, imminent threat to health and safety, as well as requests from regulators, law enforcement agencies or established incident response organizations.

B5.2 Third Party Software Vulnerability

The handling process relies on the preferred CERT, industry groups, standards bodies and other third party resources as appropriate to coordinate disclosure for vulnerability believed to impact multiple vendors (e.g. protocol design issue). Webinar's and other broad collaboration methods should be used to drive toward consensus on appropriate timeline and level of disclosure.

The handling preference for vulnerability found in an embedded third party component is direct coordination by the first party affected vendor in accordance with contractual agreements where applicable.

Product update schedule(s) outside the first party vendor's control may result in an extended timeline. The preferred CERT may be engaged as needed and in collaboration with the discoverer. A collaborative approach is also used for determination of the most appropriate disclosure threshold.

B5.3 Internal Discovery

As an industry norm, internal discovery of vulnerability affecting fielded product is handled in accordance with disclosure thresholds in effect at the time the product was most recently released.

Security issues that are fixed during a regular update cycle are documented and disclosed semi-privately as part of a release package. Private disclosure may also be effective in cases of traceable distribution of an affected version.

Internally discovered issues with severity above the product's disclosure threshold are disclosed according to the policy preferred method.

B5.4 Customer Discovery / Customer Audit Discovery

Affected business partners respect contractual agreements as currently in effect including provisions with respect to issue resolution timelines. Disclosure handling is determined based on collaboration and mutual consent.

For exceptionally high severity issues the preferred CERT, industry group, regulator or other third party may be engaged to coordinate a restricted distribution disclosure.

B5.5 Independent Researcher Discovery

The handling process for individual researcher discoveries includes extra communication to compensate for lack of established business relationship. Consensus on triage findings including severity classification is highly desirable.

The preferred timeline is a normally coordinated disclosure with the next scheduled update. However identification of additional affected products, versions, and other technical effort factors may extend the timeline. Also, in collaboration with the independent researcher, less severe issues may be assigned to a product release label commensurate with priority of other security items in the current work queue.

B5.6 "0 Day" Discovery

Public reports of 0 Day vulnerability receive expedited handling process in response to uncoordinated disclosure. If known, the Vendor reasonably attempts to collaborate with the original discoverer.

For critical severity issues, vendors may consider partial patches and other workarounds to help compensate for the vulnerability.

Effective industry wide disclosure practices may help avoid 0 day scenarios.

Appendix C – Product Security Response Team

To perform the functions outlined in this document, such as receiving incoming vulnerability reports, assessing severity, communicating with internal support and development teams, preparing and distributing remediation information, a vendor may wish to stand up a Product Security Incident Response Team, or PSIRT. [D7]

The organization of a PSIRT will vary for each vendor. An effective team will need to interact with various other functions within the vendor organization and potentially extend to the vendor's upstream and downstream software supply chain.

Appendix D - References

[D1] A Complete Guide to the Common Vulnerability Scoring System Version 2.0 (Forum of Incident Response and Security Teams - FIRST)

Cited: March 28, 2012
URL: http://www.first.org/cvss/cvss-guide

[D2] CVE Numbering Authorities MITRE Corporation (cve@mitre.org)

Cited: March 28, 2012
URL: http://cve.mitre.org/cve/cna.html

[D3] Microsoft Security Response Center Security Bulletin Severity Rating System (Revised, November 2002) Microsoft Corporation

Cited: March 28, 2012
URL: http://www.microsoft.com/technet/security/bulletin/rating.mspx

[D4] "Criticality" (Secunia's severity rating) Secunia ApS

Cited: March 28, 2012
URL: http://secunia.com/community/advisories/terminology/

[D5] SDL - Process Guidance Appendix N: SDL Security Bug Bar (Sample) (Microsoft MSDN)

Cited: March 28, 2012
URL: http://msdn.microsoft.com/en-us/library/windows/desktop/cc307404.aspx

[D6] Template: Secure Code Review Guidance and Code Priority Definitions (Microsoft SDL)

Cited: March 28, 2012
URL: http://download.microsoft.com/download/3/E/2/3E297035-2B7D-44F7-8EF6-08D4E9400EBE/Secure%20Code%20Review%20Guidance%20and%20Priority%20Definitions.docx

[D7] Computer Incident Response and Product Security (Cisco Press)

Cited: March 28, 2012
URL: http://www.ciscopress.com/bookstore/product.asp?isbn=1587052644

Appendix E – Code of Conduct

1. Do act to protect the safety, health, and welfare of the public

…the right to conduct business is granted by the public; protecting the public is a shared responsibility

2. Don't shoot the messenger

…accept all reports of defects without source prejudice or threat of persecution

3. Don't down play potential impact of a vulnerability

…fully triage vulnerability to understand root cause, affected products, and potential impact

4. Don't go it alone

…collaborate with the ICS stakeholder community on vulnerability disclosure and remediation to advance good practices

5. Do respond to vulnerabilities in all supported versions

…keep coordination authority and vulnerability discoverer informed of progress throughout the response process

6. Do know where to draw the line

…accept product life cycle issues exist; consider vulnerability disclosure limitations for legacy products and services

7. Do manage embedded vulnerability from supply chain dependencies

…manage dependencies and routinely assess supply chain security updates for applicability and compatibility

8. Do respect the public isn't fully protected until security updates are installed

…promote effective patch deployment strategies while accepting compensating measures as necessary for interim risk reduction

9. Do seek to find vulnerabilities before researchers do

…review products for similar vulnerabilities before publishing a security update

10. Do use disclosure press releases to focus on clarity and encourage appropriate remediation

…issue public statements only in an objective and truthful manner; avoid commercialism and panic related to vulnerability disclosure