Messaging, Malware and Mobile Anti-Abuse Working Group (M3AAWG) Comments on Product Security Bad Practices Guidance

Docket Number CISA-2024-0028
Comments due December 16, 2024

M³AAWG (the Messaging, Malware and Mobile Anti-Abuse Working Group, https://www.m3aawg.org/) appreciates the opportunity to respond to CISA's Request for Comment on Product Security Bad Practices Guidance, referring to "Product Security Bad Practices Guidance" document (October 16, 2024), hereafter referred to as "the draft guidance."

Introduction

The Messaging, Malware, and Mobile Anti-Abuse Working Group (M³AAWG, https://www.m3aawg.org/) appreciates the opportunity to submit these comments on the draft guidance. M³AAWG is a technology-neutral global industry association. As a working body, we focus on operational issues of internet abuse including technology, industry collaboration, and public policy. With more than 200 institutional members worldwide, we bring together stakeholders in the online community, developing best practices and cooperative approaches for fighting online abuse.

General Comments

Document Approach

We generally support the stated goals of reducing customer risk by prioritizing security throughout the product development process and discouraging the use of bad security practices, particularly where critical infrastructure and national critical functions are potentially impacted.

However, the document lacks clarity on its role and purpose in relation to other CISA publications and comments. The draft guidance does not specify who is responsible for taking action, what specific actions are required, and which level of the security management stack this document is meant to address. These elements should be clarified throughout. Merely avoiding bad practices will not be sufficient to meet security standards. Avoiding bad practices must be supplemented with industry-standard security best practices.

In addition, since CISA has previously issued advice on many of the areas covered, it would be helpful to clarify the objectives of this new draft guidance, the context for its release, and how it modifies or complements past guidance. For example, if the intent is to reinforce or summarize existing recommendations, this should be stated explicitly. Conversely, if the document introduces new recommendations or updates, those changes should be clearly highlighted.

Taking Ownership

The phrase "taking ownership of customer security outcomes" could be interpreted as conveying legal liability when signing on to support a voluntary program like this. Therefore, CISA should clearly define what "taking ownership" entails.

Readability Levels

We would like to urge CISA to ensure its advice is clear, concise, and actionable. , the draft guidance states:

> "Additionally, if a new KEV affecting the product is published in CISA's catalog, failure to issue a patch at no cost to its users in a timely manner if the KEV is exploitable in the product or failure to publicly document the presence of the vulnerability if the KEV is not exploitable in the product is dangerous and significantly elevates risk to national security, national economic security, and national public health and safety."

The advice offered is neither clear nor concise. Crucially, the sentence can be parsed and interpreted in multiple ways, which could lead to differing implementations and result in an invalid and unsafe interpretation.

Specific Comments on Proposals

Product Properties

Development in Memory Unsafe Languages (CWE-119 and related weaknesses)

The draft guidance effectively highlights several problematic product security practices, explicitly citing "C and C++" as examples of languages to avoid. It notes that "readily available alternative memory-safe languages" exist. Unfortunately, it does not provide specific examples of these alternatives.

- C and C++ are not always easily replaceable, for reasons such as performance concerns, functionality, or legacy code.
- C and C++ are not necessarily memory-unsafe if used with appropriate controls and practices. Various frameworks and tools such as static analyzers can achieve greater memory safety when using those languages.
- Misuse of pointers is not the only damaging programming practice; indeed, other poor practices can be more devastating and are not helped by moving away from C and C++.
- We also note that various higher-level languages rely on C or C++ to function, further complicating simple replacement.

Moving to Rust, Java, Python or Swift may not be easily achievable for many manufacturers; interpreters or compilers. Some platforms are simply not available, and retrofitting older devices that are still very much in use with new code is not only costly but brings its own perils. For example, improving device firmware is time-intensive and would be further complicated by a conversion to a new programming language. Extensive testing is also needed to make sure that new vulnerabilities, as well as functional errors, have not been introduced by the change.

Thus, we recommend that CISA include references to resources and solutions that address these identified issues. For instance, CISA could:

- suggest safer programming frameworks;
- provide guidance on adopting secure coding practices;
- rely on the use of scanning tools (secrets, credentials, memory, crypto) rather than only recommending specific languages that may not always be usable; and
- outline steps to implement a secure software development life cycle that includes strong testing, verification/validation, quality assurance, and DevSecOps practices.

Inclusion of User-Provided Input in SQL Query Strings (CWE-89)

The draft guidance flags the issue of unsanitized strings specifically related to SQL query strings and OS command strings.

We support the draft guidance's recommendation regarding systematic prevention of SQL injection vulnerabilities by consistently enforcing the use of parameterized queries and by delineating inputs / data and commands / code. However, we believe that readers' understanding would be enhanced if concrete examples of recommended approaches were given. Alternatively, if the aim is a more general focus on sanitization, CISA should consider

providing a more holistic discussion of the concern and relevant higher-level recommendations that include other inputs such as database lookups, APIs, and dynamically created command structures.

Presence of Default Passwords (CWE-1392 and CWE-1393)

The draft guidance also provides recommendations for avoiding default passwords. We believe that in addition to the discussion of multi-factor authentication (MFA), other alternatives to traditional passwords like digital client certificates and hardware tokens such as FIDO-certified security keys should be recommended and discussed in more detail.

While the use of digital certificates and hardware tokens may increase costs and potentially introduce new operational complexities, the additional cost may be justified for critical and sensitive applications, for data, or for both.

Crucially, dealing with the presence of default passwords is necessary but insufficient. For example, password security also requires appropriate storage, custody, and a securely designed life cycle. We note that the effort required to secure password-only approaches may be greater than the implementation of MFA or alternative methods.

Presence of Known Exploited Vulnerabilities

This section's introductory sentence, "The release of a product used in service of critical infrastructure or NCFs that, at the time of release, includes a component that contains an exploitable vulnerability present on CISA's Known Exploited Vulnerabilities (KEV) Catalog…" yields a couple of questions.

First, CISA should clarify how to define "time of release" in this case, specifically with regard to the ongoing, gradual patching of security vulnerabilities, as is common industry practice. As written, this may be interpreted to mean that only patches fixing all known vulnerabilities would be publishable. This would stop manufacturers from quickly fixing vulnerabilities.

Second, we note that not all KEVs are practically fixable (but possibly mitigatable). Some of them cannot be addressed with firmware/software updates. Thus, CISA should provide advice related to the mitigations available in difficult cases like these.

Third, we note that the proposition regarding issuing patches "at no cost to its users in a timely manner (under no circumstances longer than 30 days)" is a very expensive proposition at the very least, and possibly even impossible to achieve in some cases. The SDLC for

patches may include various critical steps such as analysis, design, implementation, testing, regression, quality assurance, and integration. While speed is important, so are the quality and testing lifecycle of the patch and the mitigation of risks associated with patching.

We support the draft guidance's recommendation that logs should be made available in a machine-readable standard format as a baseline. However, we believe that this recommendation, though necessary, is insufficient.

CISA should expand this recommendation to include and consider the risk-cost evaluation for:

- log integrity protection (e.g. WORM, which allows only writing to but not deleting or altering prior writes, or secure backup to other systems);
- centralized sysloging;
- storage concerns;
- classification;
- log filtering and verbosity/logging levels;
- use of centralized log analysis and collation (e.g. IDS/IPS);
- prioritized alerting; and
- automated detection.

Early detections on the network level may indicate compromise, and triage is simplified by having additional data sources available.

Second, logging has a considerable procedural and organizational component, such as with analysis and response, which largely remains unaddressed. We encourage CISA to recommend the creation of clear logging policies (regarding sharing, redaction, collection points, and so on), procedures and approaches, as well as better sharing of log and incident data between vendors, business partners, and other stakeholders. The lack of visibility resulting from insufficient sharing of actionable intelligence and data is a major security threat.

Presence of Open Source Software with Known Exploitable Vulnerabilities

The draft guidance's warning about software manufacturers' "failure to issue a patch or other mitigation at no cost to the product's users in a timely manner" presumes the product owner has the expertise to modify the open source software. In addition, it may take the open source community a long time to recognize and mitigate the issue. While using or building a product with open source software does not exempt the product owner from action, open source maintainers are not usually under the influence of product owners.

We note that individual downstream vendors or providers forking open source software to fix issues does come with considerable drawbacks. The concerns raised above regarding the proper process for developing patches voiced above do apply here as well. The SDLC for patches may include various critical steps, such as analysis, design, implementation, testing, regression, quality assurance, and integration. While speed is important, so are the quality of the patch and the mitigation of risks associated with patching.

Although CISA addresses this concern ("Software manufacturers should responsibly consume and sustainably contribute to the open source software that they depend on"), we emphasize that open source development is marked by its voluntary nature, often insufficient resourcing, and generally a complex ecosystem.

- While the provision of Software Bills of Materials to the software product owner is important for issue resolution, awareness, licensing, and other factors, it may be beneficial, at least in some cases, to protect these documents as proprietary, sharing them carefully with only those who incorporate this software into their systems. Abuse, malware, and threat actors may benefit from having a roadmap of dependencies and possible vulnerabilities. Advice on how to assess different publication options and approaches to sharing SBOMs would be beneficial.
- Downstream developers should always download open source software artifacts from trusted software repositories and, crucially, verify their integrity using code signing or checksum validation.
- While "run[ning] security scanning tools on each open source software component" is useful, CISA should provide information and guidance on available tools and approaches it deems sufficient.
- The guidance on updates and patching processes should be expanded. Updating from unverified sources is a security issue; we recommend expanding this guidance to include configurations (redirecting sources), executable information (log4j), and customizations that allow dynamic software deployment or software configuration. Processes, reviews, and cryptographically signed configurations are valid steps to mitigate these risks.
- While planning to update software at the outset—including to new major versions—is good practice in general, we note that this specific recommendation might be difficult to implement. First, major revisions may require complete rebuilds of applications, which comes at a significant cost, financially and otherwise. That cost is hard to estimate during planning and design. Second, it may be impossible to plan for such updates at the time of release, as feature sets and product lifetimes change through the development and product lifecycle. Third, vendors cannot plan for as yet unknown

changes to underlying code such as libraries or operating systems or services with which their product interacts. Crucially, this is also the case for security vulnerabilities, particularly those in hardware, that are impossible to fix without complete replacement, re-development, or both.

Lack of Multifactor Authentication

We agree that products should support MFA as a baseline and that admin accounts must have MFA enabled by January 1, 2026 (approximately 1 year from now).

However, we recommend discouraging the use of text messages and automated phone calls as the second factor for administrators and higher-risk applications. Those approaches are more vulnerable to interception. Instead, more advanced options like hardware and software tokens or digital certificates should be recommended.

Organizational Processes and Policies
Failing to Publish Timely CVEs with CWEs

Vulnerability disclosure should be done responsibly. While action without undue delay is central to containing and mitigating issues, immediate publication that does not also offer a fix for sufficient time to patch may be detrimental. For example, badly managed disclosure could highlight vulnerabilities for malicious actors to take advantage of, precipitating a race between the software vendor and cybercriminal adversaries.

Responsible disclosure would ensure that vendors have the opportunity to produce, test, and deliver patches for user deployment—hopefully before the exploitation becomes widespread. Judicious disclosure that balances the need for transparency with operational needs and timelines is implicit in the draft guidance's final deficiency callout, but should be made an explicit bullet point for clarity.

Failing to Publish a Vulnerability Disclosure Policy

The draft guidance recommends that software manufacturers publish a VDP "that authorizes testing by members of the public on products offered by the manufacturer."

First, we note that this wording may have to be revised to address legal and procedural concerns; for example, addressing DCMA, copyright, contractual, and liability issues.

Second, the balance between bug bounty programs, ethical disclosure, and "inviting testing" should be more clearly explained.

CISA may want to clarify who private "security researchers" are and how these experts can act in a manner consistent with ethical disclosure guidelines and practices as well as applicable laws (see first point).

Conclusion

We appreciate the opportunity to submit these comments, and we welcome further opportunities to engage as needed to answer any questions during this process. Please address any inquiries to M[3]AAWG Executive Director Amy Cadagin at comments@m3aawg.org.

Sincerely,

Amy Cadagin
Executive Director
Messaging Malware Mobile Anti-Abuse Working Group
comments@m3aawg.org
P.O. Box 9125, Brea, CA 92822