Software Assurance Tips

A product of the Software Assurance Tips Team[4]

Jon Hood

Monday 13th June, 2022

1 Don't Limit your CWEs

Updated Wednesday 11th February, 2026

One of the trends becoming popular in recent DoD contracts is a requirement for Prime contractors to scan for and fix a subset of Common Weakness Enumerations (CWEs) based on a list of what the government deems to be the most relevant to the system. Much of the language is derived from sample contract language provided by the DoD. Some sample language includes:

> The Government will provide to the Prime contractor a list of the Top-n most important CWEs. All software delivered by the Prime contractor shall be free of all of the CWEs in the Government's Top-*n* CWE list. If a CWE in the Government's Top-*n* CWE list is found to be present in the delivered software, the Prime contractor shall be liable only to repair the defect within XX-days at the contractor's expense.[5]

Including language such as this in contracts creates difficulties that the acquisition team should be aware of. Those difficulties include:

1. the possibility of a manipulative Prime developer or acquisition customer
2. the lack of prioritizing potentially critical CWEs
3. a moving-target metric that changes often

Today's Software Assurance Tip will attempt to draw attention to these pitfalls in an effort to help acquisition teams be aware of behaviors and metrics that may introduce unintended security errors for a program's software assurance process.

1.1 Manipulative Primes and Customers

Prime development teams involved in acquisition contracts often have the difficult task of balancing the desire to make their company profitable with the security of the customer they are supporting. No matter how much a company spends on security and safety, there is always something else they can be doing to improve their security stance. It's why RMF controls and cybersecurity are built into the acquisition process: the customer must define "how much is enough" security to have confidence in the delivered system.

The Top-*n* language[5] is one such "how much is enough" limiter intended to give the customer reasonable levels of assurance while limiting the liability of the Prime contractor from having to do an overly burdensome number of security verifications.

This creates a potential situation of abuse if relationships between the Prime and the Customer become strained. Since the CWEs are a hierarchy in addition to being a list, it becomes trivial to hide a buffer overflow from having to be addressed. Consider the Top 25 CWEs for 2021.[1] While operating outside of the bounds of memory (CWE-119) is 17 on the list, many tools report the issue to CWE-120 which is absent from the list. Lest the acquisition team attempt to compensate for this and add CWE-120 to their top-*n* list, the Prime can just classify them at an even lower level as stack-based or heap-based overflows (CWE-121 and CWE-122 respectively). By the time an acquisition team traverses the CWEs to include everything that *could* be an issue, their top-*n* list becomes nothing less than the entirety of the already existing CWE hierarchy.

If the contract language settles on a fixed number for *n*, the acquisition customer can create a dynamically changing list of CWEs that must be addressed. After all, as new CWEs and classes of weaknesses are released and included in the CWEs, shouldn't a program's software assurance process adapt to changing threats? In 2017, homoglyphs were introduced to the CWE list, and we have observed how they can absolutely cripple a weapons system.[2, 3] Nevertheless, I've never seen CWE-1007 on any top-*n* list! Suppose the threat landscape changes such that homographic representation issues become important. The program should absolutely have the power to include it in their list of important CWEs.

Overzealous acquisition Customers and manipulative Prime developers pose a serious and meaningful threat to this type of contract language.

1.2 Forgotten CWEs

In addition to the CWE-1007 example above, another important example that we have run into frequently is the presence of CWE-917, Expression Language Injection, which made its debut on the 2021 Top 25 list by being listed in the *Weaknesses on the Cusp* list as #30.[1] Since 2016, our software assurance team has identified this as one of the most dangerous weaknesses we run into, and the National Vulnerability Database agrees with an average CVSS score of 9.05: higher than any other CWE on the list.

A contract which implements a CWE limit must balance the **prevalence** of an issue with the **security impact** of the risk it presents. What's more dangerous for the program: ignoring 1000 resource leaks that may exist in the software or the 1 location where an attacker controls the input into a regular expression match? That answer may differ amongst varying systems.

1.3 A Moving Target

Because of the changing system and threat landscapes, a top-*n* list is unreasonable to define statically in the contract language. And because of a Prime's need to control costs, it is unreasonable to define it dynamically. This gets into the key recommendation of today's tip: for a program that feels it necessary to define a top-*n* list contractually, there should also be language to keep it updated at least annually to address the changing security concerns of the program. My recommendation is to add language like the following:

- The Top-*n* list may be changed at most yearly by the acquisition team.
- Each reported CWE must also be reported as findings against its hierarchy of CWEs as defined by the Research Concepts view (CWE-1000).

Placing a limit on how often the Top-*n* list can change prevents an abusive acquisition team that creates a constantly moving target that the Prime can never hit while still permitting reasonable updates to the security posture of the system. Requiring Prime teams to report the hierarchy of their findings prevents an issue with them hiding behind more esoteric CWEs.

References

[1] CWE Content Team. “2021 CWE Top 25 Most Dangerous Software Weaknesses”. In: (2021). URL: https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html.

[2] Jon Hood. “Homoglyphs and Homographic Attacks”. In: SwATips.com (2021). URL: https://www.swatips.com/articles/20210510.html.

[3] Jon Hood. “Malicious Injection of Source Code”. In: SwATips.com (2021). URL: https://www.swatips.com/articles/20211129.html.

[4] Jon Hood, ed. SwATips. https://www.SwATips.com/.

[5] John R. Marien and Robert A. Martin. Incorporating Software Assurance into Department of Defense Acquisitio Tech. rep. Washington, D.C.: Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2017. URL: https://rt.cto.mil/wp-content/uploads/2019/06/Incorporating-SwA-Contracts-2017-11-15.pdf.