From:	Doug Gastonguay-Goddard
To:	5GChallengeNOI
Subject:	5G Challenge Comments on Security
Date:	Tuesday, January 19, 2021 8:59:49 PM

This comment is in response to section II, Incentives and Scope, questions B and C.

II B. Could a Challenge be designed that addresses the issues raised in previous questions and also includes test and evaluation of the security of the components?

II C. Could a Challenge be designed that would require participants to leverage software bill of materials design principles in the development of components for an open 5G stack?

Security can serve as a significant differentiator for American software and hardware as we begin to compete more on the global stage. Onshoring capabilities may help ensure supply chain security, but for a sustainable economy we must land sales to other nations and allies. Reliability and security can be key selling points for American manufactured goods.

Component Design and Challenge Frameworks Friendly to Security Audits
It is critical that challenges incentivize security audit friendly design. This means components should be modular and well documented. In addition to this assisting manual audit and code review, this modularity also helps ensure that the components will lend themselves well to fuzzing. Fuzzing is the process of sending mutated inputs in an attempt to trigger bugs. It can greatly improve the performance of fuzzers to isolate and instrument components (e.g., a single parsing function).

Teams should consider integration with a continuous integration fuzzing framework, such as https://github.com/google/oss-fuzz. Critical open source software can get access to free compute time under oss-fuzz and partner programs. For components that fall outside of open source, the supported languages, or architectures that oss-fuzz supports, custom harnesses should be created.

The documentation aspect will allow security analysts to get up to speed with design considerations and security boundaries more rapidly. This is critical in time boxed assessments, as more time will then be spent on security considerations, rather than on reverse engineering and understanding code. Both of these suggestions can greatly improve the ability to find and fix bugs.

Designing a Challenge that directly requires such modular and harnessed design as core principles - rather than as an afterthought or layered approach - will enable the detailed and difficult work needed to ensure the security of the components. However, note that alone these principles will not solve or ensure the security of a component, and dedicated product security effort will likely be needed to review and verify the security.

Software + Hardware Bill of Materials
In our experience, both software (SBOM) and hardware (HBOM) bill-of-materials are critical to the security of systems.

Our tool, Pilot Security (https://www.pilot-security.com), audits embedded device firmware images. Among other features, it leverages SBOM to identify potentially vulnerable components such as outdated libraries. It also goes further to analyze reachability and accounts for the micropatching of functions. Micropatching is where a subset of the functions of a program may be patched against a vulnerability, even though the majority of the program appears to be the previous vulnerable version. These measures cut down on noise and false positives in bug reporting.

Additionally, HBOM can be used for obsolescence management. In the same way that an outdated library can indicate vulnerable code, an outdated component can indicate a greater chance of failure or security implications with respect to its firmware. Furthermore, specific hardware modules and designs convey information about *ability* to implement security to an experienced product security researcher. Lastly, we have observed on numerous occasions in our product security work how an HBOM can help rapidly assess the potential impact of a new vulnerability disclosure (e.g., a bug relating to a WiFi chipset).

Both of these areas must be tracked in an effective mitigation strategy, and automated tools are critical to make these achievable at scale and over time.

Secure Debugging
Many threat models disregard physical access as out of scope. For this reason, there are often debug ports available on devices that act as a connect-for-root. These interfaces should be secured with standard cryptographic schemes. For critical infrastructure, it must be considered that an attacker would go through lengths to gain physical access. Minimizing all attack surfaces, including the physical, is necessary for delivering secure infrastructure with confidence. Work which we contributed to under DARPA's RADICS program which enabled secure embedded debugging is directly applicable to 5G components.

Memory Safe Languages
Memory corruption vulnerabilities are still a leading class of critical vulnerabilities in software. Despite the existence of memory safe languages such as Rust and Go, people still begin projects in memory unsafe languages. Rust is well suited to high-performance low-level embedded applications. Go is an amazing language for servers and leveraging multiple cores for computation. The challenges should strongly incentivize the use of memory safe languages.

The United States cannot maintain the common paradigm of cobbling together any code that works in order to create a capability at the lowest cost. If we wish to compete on the world stage as an electronics manufacturer and solutions provider - and secure our infrastructure, our designs must be deliberate.

We look forward to seeing the direction that the 5G Challenge takes, and would welcome providing our experience in product security as appropriate to ensure that these capabilities are secure.

Thank you,
Douglas Gastonguay-Goddard
River Loop Security