SYNOPSYS®

Criteria for designating "critical software"

Modern applications are effectively highly distributed systems. While some components may resemble the monolithic application paradigms of the past, design patterns prevalent in practices like cloud-native design recognize that scalability of a system is a function of its resilience.

From the perspective of criticality, this trend ascribes a limited number of attributes to software – attributes that relate more to how software works and less the function it performs when deployed. Some of these key attributes can be described as:

1. *Owner of a source of truth*. Any source of truth is a system whose opinion to the software system is viewed as trusted. An example of a source of truth is an authentication and authorization service. If compromised in any fashion, security assumptions made elsewhere within the system become suspect. With the growth of cloud-based databases, third party APIs, and single sign-on mechanisms, the impact of a compromised source of truth could impact multiple infrastructural elements, services and systems;
2. *Gatekeeper to elevated access*. Unfortunately, development teams don't universally create their software using principles of least privilege, nor are systems universally deployed using such principles. Least privilege is tantamount to a tax on both development and operations teams who seek to maximize their output by imposing as few limitations on their respective teams as necessary. From a practical perspective, this means that operations teams often have a strong reliance on edge and perimeter defenses, such as firewalls, over practices that sandbox the application and its constituent services. While Executive Order 14028 addresses the concept of zero-trust, that concept isn't applied to the machine to machine environment present within the datacenter;
3. *Provisioner of resources*. A compromised resource provisioner can impact system operations in ways that are difficult to detect; such as overloading nodes in a compute cluster, starving systems for RAM or CPU, limiting execution times, deauthorizing access to services, and launching unapproved applications. Threat models created for provisioners must account for the impact of compromised sources of truth such as authentication services and configuration management systems, while also recognizing that a compromise of one provisioning service could impact other critical software;
4. *Auditor of service activity*. An ideal audit log for any system is one that is immutable and contains sufficient detail as to reconstruct the operational state of the system at any point in time. Unfortunately, few audit logs are truly immutable, and most were designed from the perspective of a developer seeking to debug their code. With increasingly distributed systems, particularly systems following cloud-native, microservices, or serverless paradigms where the lifespan of an application instance is limited, determining the state of any production system is challenging at best. While aggregated log analysis from services is a common element in the early warning monitoring used in a modern datacenter, that system effectively ascribes the role of "source of truth" to audit services present within the provisioning service and within the applications – likely without fully recognizing the threat profile associated with a source of truth.

Each of these four attributes can be applied to any application independent of its complexity and function within an organization, but each informs the overall risk present within the application

and the risk borne by teams operating that application. As the application scales up in usage and out in deployment complexity, any gaps in understanding related to the risks present in these attributes provide wider opening for an attacker to compromise the system in hard to identify ways.