3. Guidelines outlining security measures that shall be applied to the federal government's use of critical software

Submitted by:
Jeff Williams, Veteran Application Security Industry Leader
jeff.williams@contrastsecurity.com
https://www.linkedin.com/in/planetlevel/
410-707-1487

We write to encourage NIST to require that all critical software applications and APIs include the ability to detect attacks, create visibility into attackers and attack vectors, and can prevent common vulnerabilities from being exploited.

The average web application and API is attacked over 13,000 times every month. The overwhelming majority of these attacks are undetected. We know that applications are going to continue to go to production with serious vulnerabilities. We also know that new vulnerabilities will be discovered in libraries that are already active in deployed applications. Even worse, we know that new classes of vulnerabilities will be discovered over time that instantly cause existing production applications to become vulnerable overnight.

Historically, applications were defended at the perimeter, with proxy firewalls that analyze network traffic before it reaches software. Unfortunately, modern applications use a panoply of protocols and data formats that significantly compromises the effectiveness of this approach. For example, applications or API traffic that uses XML, JSON, or serialized objects are significantly more difficult or impossible for firewall approaches to analyze.

This inaccurate network attack detection approach has left most organizations and agencies completely blind to application layer attacks. Without application threat intelligence, defenders have no way to make informed decisions about where to improve application security.

We urge NIST to require that critical applications have the capability to detect attacks and defend themselves. This means that they must:

1. **Detect attacks and create visibility**. NIST should require that all applications detect attackers and capture details of attack attempts. The single most impactful step organizations can take to defend their applications is to simply detect attacks and block further interaction from that user. There are numerous ways to accomplish this, each with advantages and disadvantages, but attack detection can no longer remain optional. Currently, attackers can attack applications and APIs without detection or consequence until they find an exploitable weakness. Application threat intelligence is critical to feed back to development teams to help them prioritize defense efforts.

2. **Use runtime protection**. Runtime protection adds defenses to applications post-development. This approach is highly consistent with a zero trust architecture where each system defends itself. Runtime protection can take advantage of the full context of how the application uses network traffic to make much more accurate decisions about whether an attack is real or not.

Runtime protection is standardized in NIST 800-53 requirement SI-7(17), which requires the use of runtime application self-protection to protect against attacks.

3. **Disclose all incidents**. NIST should mandate that successful attacks and exploits of critical software must be disclosed. Full visibility into runtime security not only informs the public and oversight officials about the security of government software, but creates a culture of “security in sunshine” that encourages teams to produce more secure software.

4. **Continuous security testing**. Because threats change and new weaknesses and specific vulnerabilities are continually discovered, security testing is an ongoing challenge. All critical applications must be continuously tested for security vulnerability. Even third-party applications that do not include source code can be extensively tested using interactive application security testing.

5. **Require visibility**. NIST should require agencies and other organizations to disclose information about the security of the software they run. Without this disclosure, buyers, consumers, and users cannot make informed decisions about security. This is a classic market failure that disincentivizes cybersecurity in the software market. This requirement augments labels and data sheets that come with critical software with details about how the application has been deployed, configured, and run.