What You Really Need to Know

As time passes, developers find new design patterns, development frameworks, and integration and delivery strategies that change the way applications are created and maintained. This forces a list like the OWASP Top 10 to also adapt and change as new metrics are used to determine what types of vulnerabilities are associated with the highest levels of organizational risk. The final version of the OWASP Top 10 list of 2017 has some interesting changes that are important for software development teams to be aware of.

Two of the new additions to the list, **A4: XML External Entities** and **A8: Insecure Deserialization**, have been issues that have been known throughout the security industry for years. But as research and awareness of how potentially damaging these vulnerabilities could be has increased, it became clear that they needed to be highlighted by including them in this most recent version of the OWASP Top 10 List. The prevalence of XML functionality and the serialization of data objects forces developers to be aware of the potential pitfalls that could occur if these features are not carefully and cautiously defended by focusing on writing secure code and then testing these features thoroughly.

But also, as time has passed, our technologies have changed for the better. Modern browser technology and security features within development frameworks has made **Cross Site Request Forgery** much more difficult to exploit. This has led to its removal from the OWASP Top 10 list along with **Unvalidated Redirects and Forwards**. Even Cross Site Scripting, which in 2007 was #1 on the OWASP Top 10 list, has now been downgraded to #7 due to the education of developers and testers on how to combat this notorious vulnerability. But even though Cross Site Scripting has been downgraded, developers still must remain vigilant in defending their users from becoming victims of this issue.

Even though we have had success in reducing the risk that some former OWASP Top 10 vulnerabilities appearing in applications, there is still a lot of work to do. **Command Injection vulnerabilities remain #1 on the OWASP Top 10 list**, just as they were in the previous list of 2013. Developers are still not defending the interpreters within their environments from malicious inputs. This may be due to the prevalence of interpreted languages like SQL, Javascript, PHP, Perl, etc. AND developers believing that their interpreters are well hidden from the users. But this belief in "security by obscurity" gives them a false sense that their interpreters are protected. So, the struggle to defend against Command Injection continues. Maybe ten years from now, and lots of education and training, this will become less of a concern….and there will be a different vulnerability to be more concerned about.

2017 OWASP Top 10

1. Injection
Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the inter-preter into executing unintended commands or accessing data without proper authorization.

2. Broken Authentication
Application functions related to authentication and session management are often implement-ed incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

3. Sensitive Data Exposure
Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

4. XML External Entities (XXE)
Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks. Restrictions on what authenticated users are allowed to do are often not properly enforced.

5. Broken Access Control
Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6. Security Misconfiguration
Security misconfiguration is the most commonly seen issue. This is commonly a result of inse-cure default configurations, incomplete or ad hoc configurations, open cloud storage, miscon-figured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

7. Cross-Site Scripting (XSS)
XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

8. Insecure Deserialization
Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

continued...

9. Using Components with Known Vulnerabilities
Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

10. Insufficient Logging & Monitoring
Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

SECURITY INNOVATION