SYNOPSYS®

Who Needs OWASP?

Create Your Own Top 10 List

TABLE OF CONTENTS

Introduction

A list of critical web application security vulnerabilities is a necessary risk management tool.[1] Equally true is that each organization has a different set of vulnerabilities plaguing their applications. To complete a trifecta of fundamental truths, crowdsourced lists such as the OWASP Top 10 rarely reflect an individual organization's priorities. Given all that, many organizations continue to download the OWASP Top 10 and try to use it to guide their software security efforts. Since this likely won't achieve the desired result, why not use it as inspiration to create your own evidence-based, customized list?

This document guides you through the process of cataloging your firm's most common web application security vulnerabilities. But, before identifying those vulnerabilities, we must answer one very important question.

Why is your vulnerability list so important?

Identifying the vulnerabilities most prevalent in your organization's applications provides real data. Data that encourages your development and security teams to actively improve their skills and processes. However, creating a satisfactory Top N list requires more than simply sorting one day's or one application's bug data.

Building a list from the results of code review, various security testing methods, and actual incidents experienced by your firm and/or industry allow you to craft a holistic vulnerability list. You'll then be better prepared to drive change that reduces risk in your organization.

> Simply sorting the day's bug data by number of occurrences does not produce a satisfactory Top N list.

Customize your own comprehensive Top N list in 4 steps

Step 1: Gather vulnerability data

To understand what you're up against, you'll first need to gather web application vulnerability data for a given period. Go back as far as you can with the technologies you have. Six months of data should be a minimum benchmark. Obtaining such data is easier to gather if your firm tracks security bugs in a central repository.

If no repository currently exists, identify sources that contain vulnerability data. Collect reports associated with these sources. Next, extract the vulnerabilities into a spreadsheet or another data store.

1 By "critical" we mean "most undesirable." By "vulnerabilities" we mean "vulnerability types," not individual security defects.

Step 2: Normalize the vulnerability data

Normalizing vulnerability data ensures that each vulnerability instance is only counted once per application. You don't want one bug-filled application to skew your results. A step-by-step approach to achieve this involves the following steps:

1. **Remove all re-test data from the data set.** If the first test identifies a vulnerability that is not fixed by the time a re-test takes place, list the vulnerability only once.
2. **Correlate penetration testing, incident response, and code review findings** to remove multiple vulnerability instances. The instances may have been identified using different methodologies even though the same bug caused them. For instance, a cross-site scripting (XSS) vulnerability could be discovered during source code review, exploited during penetration testing, and abused by a real attacker. You should still list this vulnerability only once.
3. **Normalize finding names.** For instance, the findings identified as "Stored XSS on admin page" and "Stored XSS on help page" could be renamed to read "Stored XSS." The granularity of name identification determines the results of your Top N list. Use your judgement to determine what works best for your organization.

Combining and harmonizing results from many kinds of defect discovery methods results in a richer assessment data set. Thus, more accurately reflecting what might be present in production environments.

Step 3: Create a vulnerability frequency table

Using the normalized vulnerability data, create a frequency table to list the number of times a given vulnerability was identified. For example, unsafe use of user-supplied data was identified 100 times throughout 150 applications over the past six months. This frequency table can help predict the likelihood that similar applications have the same vulnerability.

Again, simply sorting a day's bug data by number of occurrences doesn't produce a satisfactory Top N list. Data changes often due to different testers, tools, processes, and applications. A historical view is important. Even when you have good frequency data, the point is to enable good risk management decisions. That requires additional knowledge.

Step 4: Consider additional risk factors

Once you have vulnerability frequency data, add analysis for three important risk factors:

1. **Detectability.** The likelihood an attacker can discover the vulnerability.
2. **Exploitability.** The likelihood an attacker can exploit that vulnerability.
3. **Impact.** The combined technical and business impact if the vulnerability is successfully exploited.

For each vulnerability in your list, add a score for each risk factor on a scale from 1-3 (with 3 being the higher end of the scale).

Determining values for these factors are often based on the discovery method of the vulnerabilities. For instance, a vulnerability exploited in the wild (e.g., originating from the incident response report) might merit a higher exploitability than a vulnerability discovered during code review. Likewise, a vulnerability detected during a penetration test likely merits a higher detectability than a vulnerability detected during code review. Don't hesitate to make adjustments based on professional opinion. Network location (e.g., Internet-facing versus internal), architecture (e.g., mobile versus thick-client), and other such criteria may be useful when assigning risk factor values.

Based on the scores from this exercise, you can prepare your Top N list. This list is now customized to fit your organization and its software portfolio. This is one of the most valuable tools your firm can have for software security risk management.

Moving forward

Your organization's software security group (SSG) should periodically update the list and publish a Most Wanted Report. This calls attention to the most vulnerable areas of your firm's software and applications. Incorporate this list into internal training material.

When shown organization-specific vulnerability data, developers are more likely to understand how the material is relevant to their work. They are also better prepared to know when and how to apply what they've learned from training.

Your Top N list may also open the door to executive attention regarding your firm's vulnerabilities. These questions will help your teams obtain the resources they need to close gaps and mature the existing software security strategy.

Be sure to adapt your approach at regular intervals to keep your Top N list up-to-date. Don't simply base your security program on a static list. Sample your own data and turn the results into a powerful risk management tool.

SYNOPSYS®

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com