Check Point
SOFTWARE TECHNOLOGIES LTD

SECURE YOUR EVERYTHING™

DEFENDING AGAINST THE OWASP TOP 10

Introduction

Protecting your web applications is critical. The lack of tightened security could lead to data breaches, downed systems, or worse. Understanding which web vulnerabilities pose the most risk to your organization, and taking steps to secure them, should be your organization's top priority.

The OWASP Top Ten represents a widely-accepted consensus on the most critical software application security flaws according to security experts from around the world. They provide a list of general vulnerability classes so security professionals can assess the level of coverage that can be achieved with their security products.

In this paper, you'll find real world examples of the OWASP Top Ten and learn how Check Point Software secures against all ten vulnerabilities.

> " Using the OWASP Top 10 is the most effective first step towards securing your web applications."

Injection

Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query. For example, in an SQL injection attack, if a form expects a plaintext username or password, an attacker could enter a SQL database code. If the form input is not secured, the SQL code would unknowingly be executed. Other examples include SQL, NoSQL, OS, and LDAP injections. SQL and Command Injection Attacks are blocked by looking for keywords.

> " SQL injection attacks represent two-thirds of all web app attacks."
>
> – Jai Vijayan

Real World Example

The Panama Papers incident resulted in 11.5 million leaked records detailing financial and attorney-client information.[1] Because there were a large number of potential attack vectors, it's not clear what method the attackers used to leak the data. However, researchers discovered that the Panamanian law firm's website was using a version of Drupal riddled with at least 25 vulnerabilities, including an SQL injection flaw[2] that allowed anyone to remotely execute arbitrary commands.

[1] "What are the Panama Papers? A guide to history's biggest data leak," by Luke Harding, The Guardian, April 5, 2016.

[2] "The security flaws at the heart of the Panama papers," by James Temperton and Matt Burgess, Wired, April 6, 2016.

Broken Authentication

Weaknesses in authentication systems, or login systems, allows hackers to compromise passwords or other data they otherwise shouldn't have been authorized to view. For example, without the proper security protocols in place, attackers can brute-force into accounts containing sensitive data or critical privileges.

Real World Example

A large e-commerce software business had faulty logic in its web app, allowing attackers to bypass authentication and grant themselves 'collaborator' access to any store.[3] Essentially, an attacker could create two partner accounts with the same business e-mail to gain full permissions to the store. The organization gave a $20,000 reward to the person who revealed the bug.

Sensitive Data Exposure

Sensitive data exposure occurs when a web application fails to protect sensitive data, such as e-mail addresses, names, banking information, and more. An API that doesn't properly encrypt sensitive data can lead to the risk of compromise. One popular method for stealing sensitive information is a man-in-the-middle (MITM) attack which allows the attacker to see the transmitted data (such as a password) in plaintext.

Real World Example

Google Project Zero discovered an issue in a major CDN provider's edge servers that leaked user passwords and other sensitive information to thousands of websites.[4] The vulnerability occurred because of the way in which broken html tags were parsed, which returned data from a random portion of the server's memory.

[3] "Shopify admin authentication bypass using partners.shopify.com," HackerOne, September 28, 2017.

[4] "Cloudflare Reverse Proxies are Dumping Uninitialized Memory," Chromium, February 19, 2017.

> " XXE is not a flaw in XML that can be patched... Ultimately it is an attack largely dependent on human error, and that means it's here to stay."
>
> – Trenton Gordon

XML External Entities (XXE)

This is an attack against applications that parse XML input. More specifically, this attack occurs when XML input referencing an external entity is processed by a poorly-configured XML parser. As a result, attackers can read sensitive data, share files, and launch Denial of Service attacks.

Real World Example

A researcher discovered an XXE vulnerability in one of the most popular social media websites.[5] While the exploit was straightforward, it allowed attackers to read local files on the target host and could pivot to other related hosts. A $10,800 bounty was rewarded for discovering the issue.

Broken Access Control

Access control refers to a system that manages access to information or functionalities. Broken access allows threat actors to sidestep authorization and perform tasks as if they were privileged users. These unauthorized actions include accessing other users' accounts, viewing sensitive files, modifying other users' data, and changing access rights.

Real World Example

Again, one of the largest social media websites in the world suffered a severe vulnerability.[6] It lacked an access control measure, resulting in the possibility of malicious users to assign admin permissions to themselves for a particular Facebook page, allowing them to post statuses and publish photos. Attackers could also deny access to the legitimate administrator or manager. The social media giant issued a $2,500 reward.

[5] "XXE on sms-be-vip.twitter.com in SXMP Processor," HackerOne, July 26, 2017.

[6] "Hacking Facebook Pages," by Laxman Muthiyah, The Zero Hack, March 2, 2021.

OWASP #6

Security Misconfiguration

Security misconfiguration is the most commonly seen vulnerability on this list. This is commonly a result of using default configurations, insecure HTTP headers, or exposed error messages that contain sensitive information.

Real World Example

The Mirai botnet executed high profile attacks through IoT devices, which was caused by misconfiguration of the passwords protecting the systems. As a result, the Mirai botnet obtained powerful DDoS capabilities, disrupting access to Amazon, Github, Netflix, Airbnb, Twitter, Reddit, and even Internet access across the country of Liberia.[7]

> " Serious access control flaws may even allow cybercriminals to access an admin console or other privileged functionality, paving the way to account manipulation, code execution, or complete system compromise."
>
> – Zbigniew Banach

[7] "Mirai Malware Targeting the Enterprise," by Dave Greenfield, Cato Networks, June 24, 2019.

Cross-Site Scripting (XSS)

Cross-site scripting (XSS) is a very common security issue found in web applications. If a website takes text input, such as a profile or status update, attackers can introduce a payload written in HTML and JavaScript. This vulnerability allow an attacker to execute script(s) in a victim's browser to hijack user sessions, alter the web page contents, or redirect the user to a malicious site.

Real World Example

A simple XSS hack was discovered on a video game platform,[8] which allowed hackers to spread malware to anyone who visited their profile pages. For example, simply visiting a gamer's profile page could download a ransomware program or direct the user to a phishing site.

Insecure Deserialization

In this threat, the attacker targets web applications which serialize and deserialize data, which can result in serious consequences such as remote code executions attacks and DDoS attacks. Serialization is the process of translating an object into a format which can be stored to a disk or streamed.

Deserialization is the opposite: converting serialized data from a file, stream or network socket into an object that the application can use. In addition to remote code execution attacks and DDoS attacks, deserialization flaws can also be used to perform replay attacks, injection attacks, and privilege escalation attacks.

Real World Example

An insecure deserialization attack caused one of the worst data breaches of an credit reporting agency, exposing the data of over 140 million US consumers.[9] What caused this breach? An insecure Apache Struts framework in a Java web app allowed execution of arbitrary code on the web servers.

[8] "A nasty vulnerability in Steam profiles potentially lets hackers spread malware (Update: It's fixed)," by Matthew Hughes, The Next Web, February 7, 2017.

[9] "The Consequences of Insecure Deserialization: The Equifax Breach," Kiuwan, August 15th, 2019.

Using Components with Known Vulnerabilities

Web developers often use components such as libraries and frameworks in their web applications, which help provide needed functionality and avoid redundant work. However, if any of these components have security vulnerabilities, then the entire app is at risk. Attackers search for vulnerabilities in these components in order to launch an attack.

Real World Example

A WordPress vulnerability was used to deface 1.5 million webpages. Attackers exploited an unpatched REST API vulnerability that allowed code to be injected into WordPress sites.[10]

Insufficient Logging & Monitoring

The key to cyber security is visibility and responsible monitoring. However, the average time to detect a data breach is over 200 days, which is staggering. This allows hackers to wreak havoc long before there is any security response.

Real World Example

One of the most significant logging and monitoring failures occurred when a data breach affected more than 500 million customers of a major hotel chain.[11] Attackers had been accessing the vulnerable network for over four years before the data breach was eventually reported.

[10] "Recent WordPress vulnerability used to deface 1.5 million pages," by Lucian Constantin, PCWorld, February 10,2017.

[11] "Marriott discloses massive data breach impacting up to 500 million guests," Washington Post, November 30, 2018.

Protect Your Web Applications

Your web applications are evolving faster than ever. As a security leader, you need to reduce risk by minimizing your applications' attack surface. With CloudGuard AppSec from Check Point, you can stop OWASP Top Ten attacks, prevent bot attacks, and stop any malicious interaction with your applications and APIs across any environment.

Ninety percent of CloudGuard AppSec customers run our solution in prevent mode. With continuous learning, your app will remain protected even as DevOps releases new content. With automated web application and API protection, you'll remain confident in your application threat prevention.

From implementation through runtime, CloudGuard AppSec automatically analyzes every user, transaction, and URL to create a risk score to stop attacks without creating false positives.

You'll also receive protection against:

- Site defacing
- Information leakage
- User session hijacking
- And more

Click here to learn more about CloudGuard AppSec.

Worldwide Headquarters
5 Ha'Solelim Street, Tel Aviv 67897, Israel | Tel: 972-3-753-4555 | Fax: 972-3-624-1100 | Email: info@checkpoint.com

U.S. Headquarters
959 Skyway Road, Suite 300, San Carlos, CA 94070 | Tel: 800-429-4391; 650-628-2000 | Fax: 650-654-4233

www.checkpoint.com