Michael Anthony Borchardt
Web Application Security: Vulnerabilities – Part One

January 15th, 2024

Objective and Scope

The purpose of this Web Application Security document is to establish a baseline understanding of vulnerabilities facing web applications in today's online environment.

Oftentimes, how to begin securing a web application is a misunderstood practice. While being drowned in technical terms and information is common, I attempt to bridge the gap between simplicity and professional security standards.

This document is intended for personnel developing and supporting web services, websites, and web applications.

1. Introduction

The materials presented within the contents of this document are collected from various organizations across the Internet. These organizations include, but are not limited to, the Open Web Application Security Project (OWASP), the National Institute of Standards and Technology (NIST), the SANS Institute, and other recognized sources of industry best practices.

This document is divided into multiple sections that cover the following topics:

Section	Description
Vulnerability Assessment	Brief description of a vulnerability assessment within context to web application security.
Vulnerability Assessment Objectives	Objectives of a vulnerability assessment.
Common Vulnerabilities in Web Applications	A discussion of the most common vulnerabilities in web applications.
Vulnerabilities in Web Server Applications	Consideration of the web server applications used to host the web application.
Vulnerabilities From Disclosure of Sensitive Information	Thinking about data policies and the disclosure of sensitive information.

In order to receive the most benefit from this document, I do recommend that you read each portion, in depth, to better understand each section.

2. Vulnerability Assessment

The goal of a vulnerability assessment, with regards to web applications, is to better understand the attack surface of the web application. A vulnerability assessment is non-intrusive. This is understood to communicate that the tester is not going to exploit the vulnerability, but the tester is going to document that the vulnerability exists.

For example, a common vulnerability documented by testers of web applications is failure to update the operating system of the server the web application is hosted on. While many web applications have migrated to use third-party cloud hosts in recent years, they’re still a number of web applications hosted on physical servers stored on site.

In my opinion, a vulnerability assessment is a report that documents all known vulnerabilities of a web application. Potential exploitations for the known vulnerabilities would also be documented by the tester. Moreover, this would allow the organization an opportunity to reduce the potential attack surface of the web application.

3. Vulnerability Assessment Objectives

Oftentimes, a vulnerability assessment has multiple objectives. With regards to web applications, the primary objective of a vulnerability assessment is to map points of potential exploitation in the application.

In addition, we have a list of secondary objectives with web applications. Secondary objectives of a vulnerability assessment include, but are not limited to, reducing server down time, ensuring the confidentiality of the data in use by the application, maintaining the availability of the data for use by the application, and adhering to standards of data collection and use set forth by the local, national, and international governments.

As a quick caveat, data collection policies vary by region. Each governing body, or jurisdiction, has policies on how to appropriately collect data, when data can be collected and used by a web application, and how to go about doing this. Whether or not these laws apply to use is dependent upon the reach of the web application. For example, a web application that interacts with EU citizens falls under jurisdiction of the EU, and, therefore, must adhere to all data collection policies enacted by the EU.

4. Common Vulnerabilities in Web Applications.

The most common vulnerability, in modern era, of web applications is the failure to update the operating system of the machine the web server is running on. While I don’t have statistics to support this claim, I am speaking from personal experience. Failure to update the operating system of the machine has opened the attack surface of more applications than I am able to count.

Moreover, if you take away one piece of information from this document, then understand that as a web application administrator you’re responsible for ensuring that the operating system be updated when updates become available. Oftentimes, many system administrators push updates off until a further date. As to why this is seen as an option, I have no idea.

Zero day vulnerabilities refers to a vulnerability that is unknown to the vendor of an application, which, in turn, makes the vulnerability unknown to the user. While this may seem like quite a rare problem, this is becoming more common than a person may think. Updating operating system software is incredibly important because of zero day vulnerabilities. Many patches are released and designed to fix multiple issues. However, if the patches are not tested appropriately, then sometimes these patches result in the creation of zero day vulnerabilities.

The second most common vulnerability in web applications is actually not the web application itself, but the services running on the machine hosting the web

application. While many system administrators take care to disable extraneous services running on a machine dedicated to hosting a web application, oftentimes, many of these system administrators fail to take this precautionary measure.

Extraneous services and applications running on a machine dedicated to hosting a web application are a serious issue. These services and applications offer a pathway into the machine. Furthermore, these services and applications also reduce the functionality of the web application, as these are processes consuming more resources. The resources consumed by these processes could be allocated to producing a more effective web application.

However, if you must run other applications on a machine hosting a web application, then be conscious of updating applications used by the machine. Scheduling downtime for updates is so important. The longer a system administrator pushes off updates for, the larger a hole in the attack surface of a web application becomes.

Although communication between web application administrators and patrons of the application is important, the point of this communication should not be obfuscated. Users of an application should be able to contact a web application administrator for a number of reasons, but this communication should not be so personal that a web application administrator is providing users with personal information. For example, I have observed many web application administrators provide their organization email address with the associated domain as a point of contact for the hosted web application (e.g. eve.hobs@pear.com). This is a significant mistake.

For a number of reasons, this is something I wouldn't do. Not only does this provide users with the schema for domain-based email addresses at this organization, but this is also provides bad actors with a name and contact information for the web application administrator. While this may seem innocuous, this information can be of significant value to a bad actor.

How, then, will users of a site contact a web application administrator? I would provide an email with a generic title associated with the domain to provide users with a point of contact. For example, help@pear.com would be an appropriate handle for an email address that users of the web application at this domain could use to contact the system administrator. This does not provide users with information they do not already have. Also, this does not provide bad actors with personal information of the web application administrator.

Finally, we're at an appropriate point to mention poor site development. While I perceive SQL injection and cross-site scripting (XSS) as a late 2000's problem, I do not perceive denial-of-service attacks as such. Most of these attacks are associated with the poor configuration, or construction, of a web application.

SQL injection is where the user will type SQL commands into an HTML form to try and get the database to behave in specific ways. Oftentimes, the user is try to gain access to the data stored in the database.

XSS is where the end-user of an application will enter a few lines of script into the application to encourage the application to behave in an unintended manner. Oftentimes, this is the result of poor application development. It is important that a web application validate all scripts before executing them.

Also included with poor web application development is denial of service attacks. A DoS attack is the process of overloading a web application with requests until the service is no longer running as intended. While I perceive this as a more advanced attack, this is still a possibility for most web applications. This attack is highly dependent upon the services being served by the web application and the sophistication of the attacker.

They’re an endless number of vulnerabilities in web applications. However, my effort in this last section was to name a few of the more common vulnerabilities.

5. Vulnerabilities in Web Server Applications

Web servers applications are applications whose primary function is to serve a web application. To name a few of the most common web server applications, I would list Ngynx and Apache. Both of these web server applications are very commonly found on Linux operating systems, and I understand both of these web server applications require discrete attention to detail for secure configuration.

There is what I would reference as a “laundry list” associated with web server applications. As is appropriate, the first item on this list is to patch and upgrade the web server application. While many web server applications come upgraded upon download, this is not always the case. I would run a check on the web server application to ensure that no upgrades are missing.

Next, I would remove or disable any unnecessary services, applications, and sample content from the web server application. Applications running on a web server application offer a distinct avenue for entry by a bad actor.

Configuring the web server application for user authentication and access controls is also important. Users should authenticate through password protected methods. Also, users should be defined certain roles on the web server application, as each user should not be appointed more access than required by their job description. Roles help distinguish a clear separation of responsibilities, and, therefore, help an administrator of a web server application assign privileges to each user.

6. Vulnerabilities From Disclosure of Sensitive Information

Organization web applications, websites, and web services are often the first place a bad actor would look for sensitive information that could be used to exploit a hole in the attack surface of an organization's public web domain. While I understand its important for an organization to communicate with their shareholders, constituents, and customers, I do perceive not releasing too much information as equally important.

Publishing information online should be a carefully thought out event. An organization should consider whether or not information published via public websites or web applications could be used in future cyber attacks. Classified or proprietary business information should not ever be published on a public facing website, web application, or web service. Furthermore, personnel should be cautious of publishing this type of data on machines hosting web server applications.

Oftentimes, I've seen organizations work out what is known as a data policy. Every data policy is unique to the organization, but, in short, the data policy discusses who owns the data, how the data will be stored, and who is responsible for keeping the data safe. In many situations, an organization will define that specific sets of data cannot be kept on the same machine that is hosting a public website, web application, or web service.

For example, medical information for customers of a local medical clinic shouldn't be published on the same machine hosting the clinic's public website. Foremost, sensitive data, such as medical information, should be stored on a server behind a firewall. Second, this should not be the same machine that is hosting a public website. Oftentimes, I've seen IT personnel declare what is known as a DMZ (demilitarized zone), which is where the machine hosting the public website, web application, or web service is kept. Between the DMZ and the sensitive data is a firewall. If there needs to be some form of communication between the data and the machine within the DMZ, then this is configured carefully and appropriately.