MODULE 4C: AUDITING WEB SERVERS AND WEB APPLICATIONS

OBJECTIVES

- How to audit a web server
- How to audit a web application

WEB AUDITING ESSENTIALS

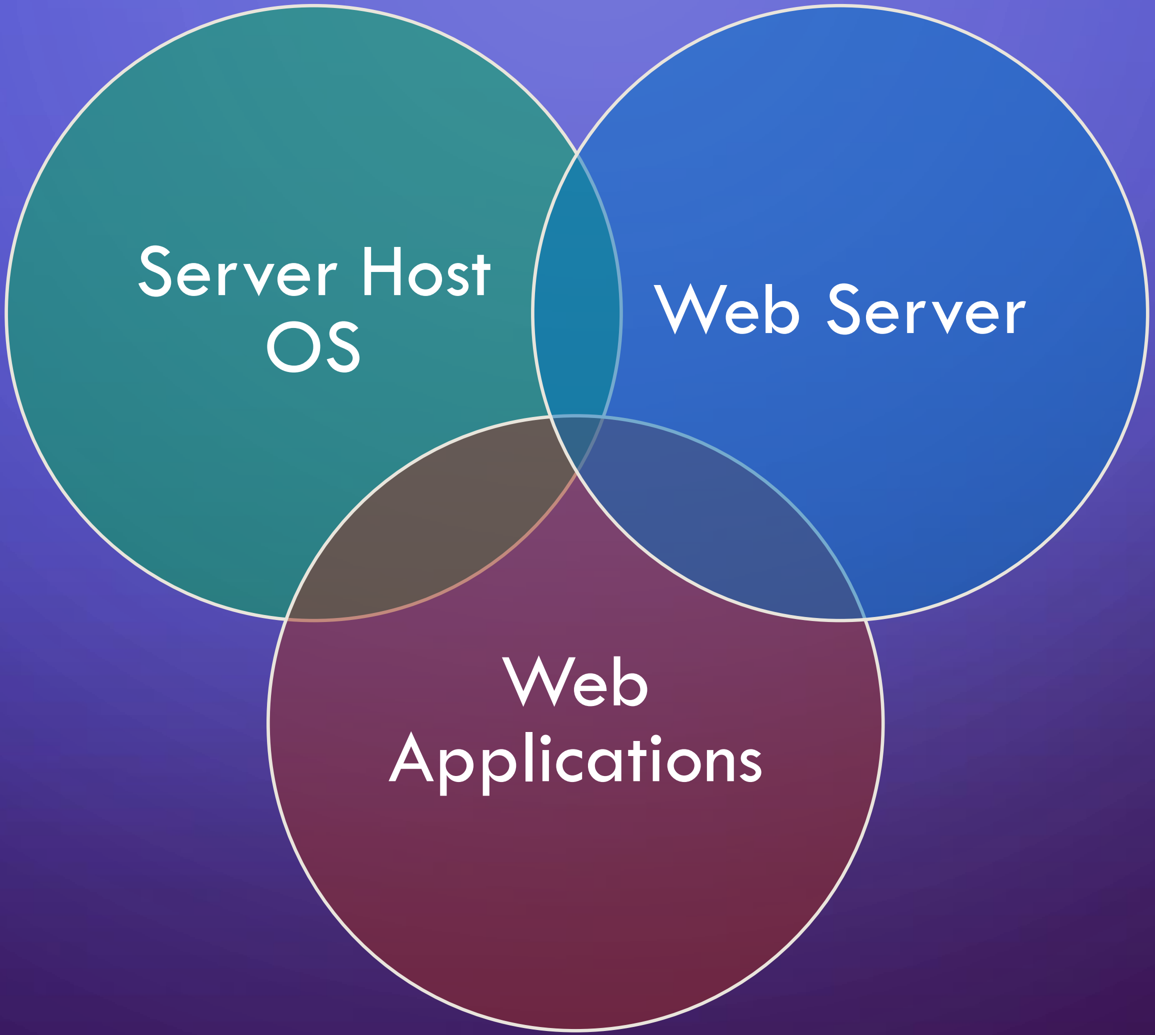
- Web servers are common targets. They can be difficult to secure, and they often contain or are used to access company secrets, personal information, or financial data such as credit card records.
- Verizon’s 2018 Data Breach Investigations Report identified web application attacks as a component in nearly 20 percent of successful company breaches. In addition, Verizon noted over 23,000 incidents in the 2018 report where a compromised web application was used for some other malicious purpose, such as spam or phishing, or was repurposed as part of an attack against another organization.

WEB AUDITING ESSENTIALS

- Like much of auditing, assessing a web server is not an exact science. Many different types of web servers are in use (servers running on Apache, nginx, or Microsoft Internet Information Services are the most common) and millions of different applications rely on web services.

WEB AUDIT COMPONENTS

Windows,
Linux,
Unix



Apache,
Nginx, IIS

AUDITING THE HOST OPERATING SYSTEM

- Servers are going to typically run on some version of Linux (Apache) or Windows (IIS).
- Auditing the host OS is important as it's underlying the web server software and any web applications.
- If there are vulnerabilities in the host OS, the web server itself is obviously vulnerable

AUDITING WEB SERVERS

- Once the operating system controls are in order, the web server layer should be examined. The basic steps provided here apply to most web server environments, regardless of the underlying operating system platform or the specifics of the web application itself. Most of the audit steps involve interviews or discussions with application or server administrators.

AUDITING WEB SERVERS

- **Verify that the web server is running on a dedicated logical system not shared with other critical applications.**
  - Identify and discuss each application with the administrator. Carefully consider the legitimate business need to allow other applications to remain on the same host as the web server. If these applications must coexist, consider bringing each of the additional applications into the scope of the audit.
- **Verify that the web server is fully patched and updated with the latest approved code**
  - Every organization has its own patch management systems and policies. For Apache systems on Unix platforms, you can use the command httpd –v to see the Apache version in use.
  - There might be a reason a webserver isn't patched though! Think about legacy systems!

AUDITING WEB SERVERS

- Verify that unnecessary services, modules, objects, and APIs are removed or disabled. Running services and modules should be operating under the least privileged accounts.
- Verify that only appropriate protocols and ports are allowed to access the web server.
  - Web servers shouldn't have email ports open? Most will likely have SSH open though

WEB SERVER NETWORK DEFENSES

- Web application firewall (WAF): A WAF is an application-aware firewall that examines web requests for malicious content. A WAF can be installed as a stand-alone device, as a feature in other firewalls, as an add-on module for a web server, or as a third-party, network-based service. If a WAF is used, you should examine how it is maintained and how alerts from the WAF are handled by the organization.
- Reverse proxy: A reverse proxy is a special kind of intermediary web server that processes requests from the outside and directs them appropriately. Reverse proxies can be used as firewalls to enable very limited traffic patterns to the target web system. If a reverse proxy is used, you should work with the administrator to understand how it is configured and what additional protections are offered. If the system is merely a pass-through, it is effectively providing no protection.
- Denial-of-service (DoS) prevention: In a DoS attack, a system is overloaded with requests, causing a crash or other unpredictable behavior. While DoS attacks can be difficult to defend at the server itself, network architectures, including rate limiting, load balancing, and other defenses, can be deployed to reduce the likelihood or impact of a DoS condition. Discuss this with the network administrators. DoS protection may be applied strategically to critical applications rather than in an across-the-board fashion.

AUDITING WEB APPLICATIONS

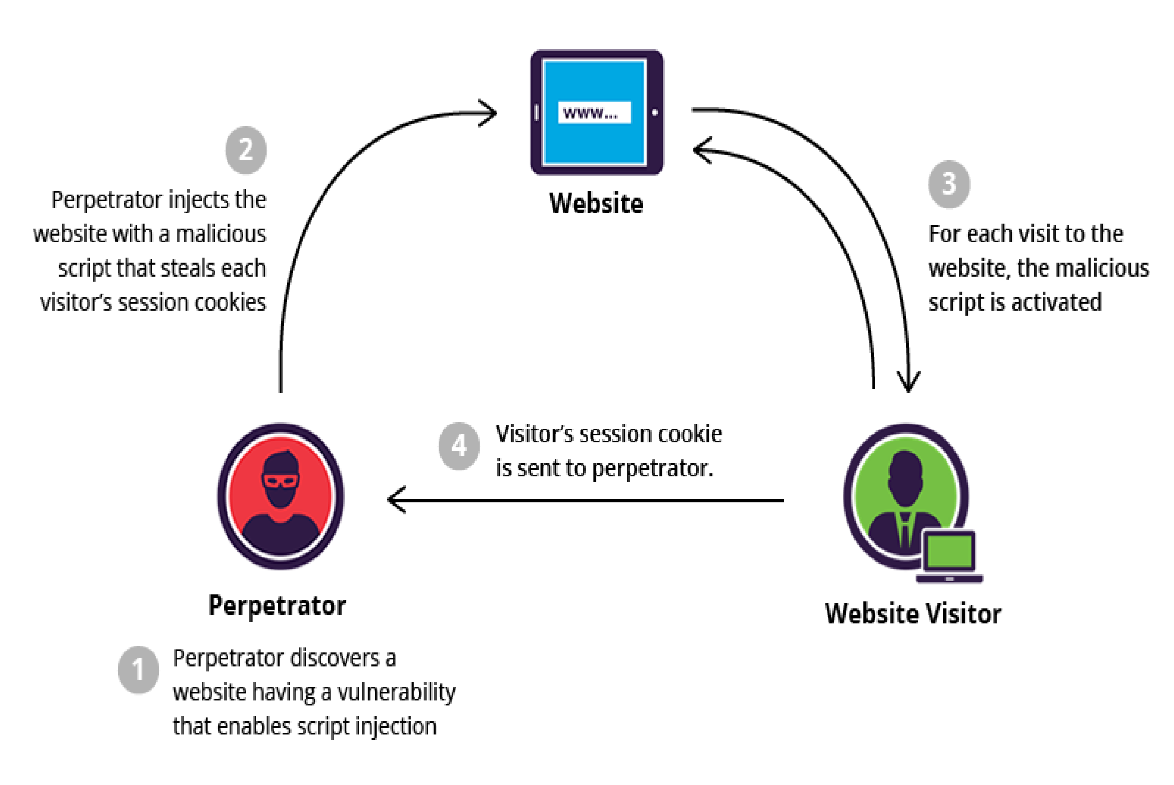
- Open Web Application Security Project (OWASP) Top Ten web application security risks.
- https://owasp.org/www-project-top-ten/

- A1:2017-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 interpreter into executing unintended commands or accessing data without proper authorization.
- A2:2017-Broken Authentication: Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
- A3:2017-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.
- A4:2017-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.
- A5:2017-Broken Access Control: Restrictions on what authenticated users are allowed to do are often not properly enforced. 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.

- A6:2017-Security Misconfiguration: Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured 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/upgraded in a timely fashion.
- A7:2017-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.
- A8:2017-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.
- A9:2017-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.
- A10:2017-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.

CROSS-SITE SCRIPTING (XSS)

https://www.hacksplaining.com/exercises/xss-stored

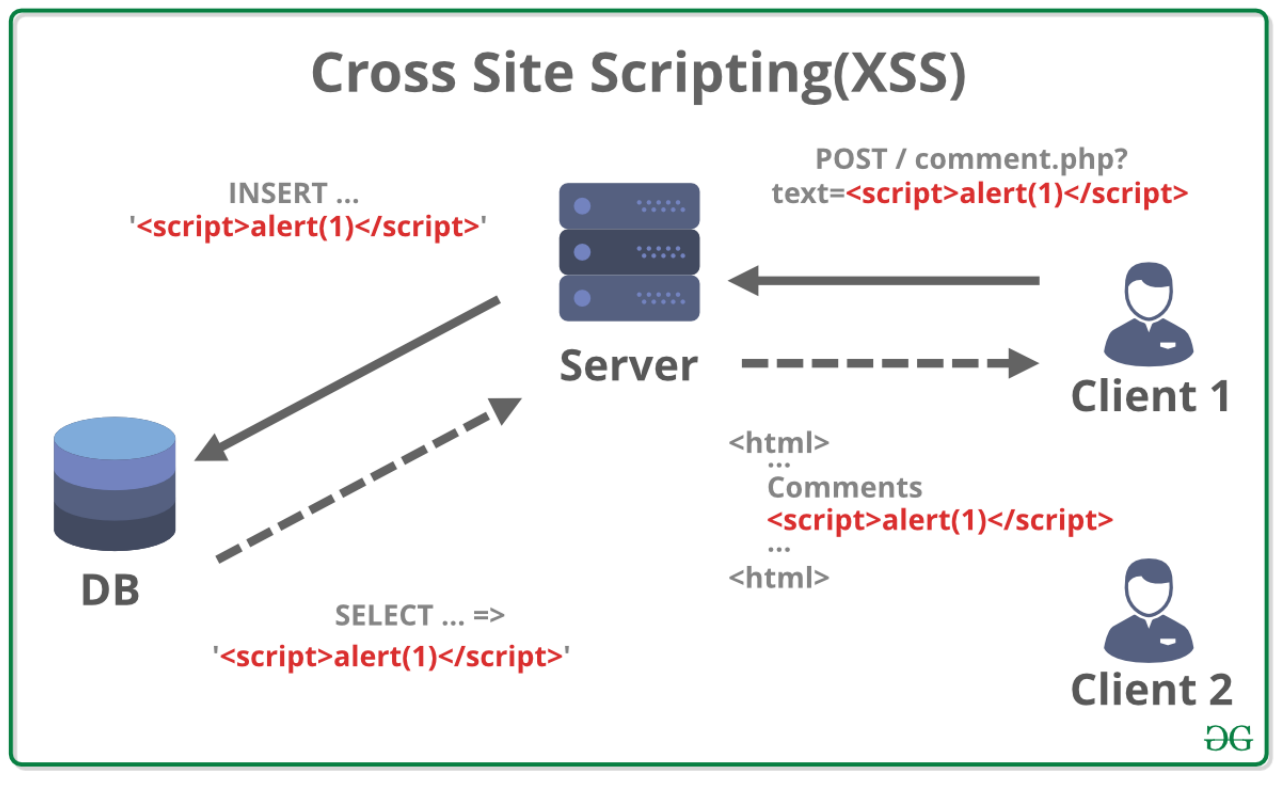


1 Perpetrator discovers a website having a vulnerability that enables script injection

2 Perpetrator injects the website with a malicious script that steals each visitor's session cookies

3 For each visit to the website, the malicious script is activated

4 Visitor's session cookie is sent to perpetrator.



INSERT ... '<script>alert(1)</script>'

POST / comment.php? text=<script>alert(1)</script>

SELECT ... => '<script>alert(1)</script>'

```
<html>
 ...
 Comments
 <script>alert(1)</script>
 ...
<html>
```