Pentest Applicatif

Sécurité Web

Cross Site Scripting - XSS

Type: client

Description

Injection dans l’url ou dans un formulaire de contenu HTML qui sera interprété par la page Web

Exemple d’utilisation

Affichage du cookie d’utilisateurs au travers de l’URL :

```
GET /xss.php?name=<script>alert(document.cookie);</script> HTTP/1.0
Host: www.example.com
```

Réfléchie

La XSS Réfléchie sera essentiellement faite au niveau de l’URL.

Un paramètre GET ou POST de l’application sera directement affichée sur la page.

Stockée

La XSS Stockée sera persistante.

Un utilisateur pourra par exemple envoyer un commentaire pouvant être lu par tous les autres utilisateurs avec du code javascript pour récupérer leurs cookies.

File Inclusion

Type: serveur

Description

Interprétation côté serveur d’un fichier arbitraire

Exemple d’utilisation

Inclusion du fichier /etc/passwd :

```
GET /lfi.php?page=../../../../../../etc/passwd HTTP/1.0
Host: www.example.com
```

Local File Inclusion

Inclure un fichier interne au serveur.

On peut ainsi :

- Lire des fichiers restreints
- Exécuter des pages non accessibles
- Exécuter des commandes sur le serveur

Remote File Inclusion

Inclure un fichier externe sur le serveur.
Le but ici est de présenter le fichier à faire exécuter au serveur.

L’attaquant peut grâce à cette vulnérabilité exécuter du code sur la machine.

[No]SQL Injection

Type: serveur

Description

Modification de la requête SQL ou NoSQL normalement exécutée côté serveur.

Exemple d’utilisation

Exploitation d'un champ en POST

```
POST /login.php HTTP/1.0
Host: www.example.com
Content-Type: application/x-form-www-urlencoded; charset=utf8

username=‘ or 1=1 -- &password=test
```

Normal

Une SQL Injection normale avec un retour, vous permet de récupérer directement le retour de la requête.

Error Based

Les erreurs ne sont pas correctement gérées et vous pouvez voir un retour d’erreur qui a évalué une partie de la requête.

Blind

A l’aveugle avec seulement un retour différent en fonction du retour (Par exemple seulement une modification de code retour)

Beaucoup plus tricky.

Command Injection

Type: serveur

Description

Modifier une commande normalement exécutée pour lancer une commande arbitraire

Exemple d’utilisation

Injection dans le champ ip_address de la requête POST

```
POST /ping.php HTTP/1.0
Host: www.example.com
Content-Type: application/x-form-www-urlencoded; charset=utf8

ip_address=;id
```

Open Redirect

Type: client

Description

Une page permet une redirection vers une URL que l'on souhaite

Exemple d’utilisation

Campagne de phishing pour rediriger vers un site en particulier

```
GET /redirect.php?next_url=http://attacker.com HTTP/1.0
Host: www.example.com
```

OWASP TOP 10

A01 - 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.

A02 - Cryptographic Failures

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS).

A03 - 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.

A04 - Insecure Design

An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

A05 - 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.

A06 - Vulnerable and Outdated Components

Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Weakness Enumerations (CWEs) mapped to the included CWEs, so a default exploits/impact weight of 5.0 is used. Notable CWEs included are CWE-1104: Use of Unmaintained Third-Party Components and the two CWEs from Top 10 2013 and 2017.

A07 - Identification and Authentication Failures

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.

A08 - Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). Attackers could potentially upload their own updates to be distributed and run on all installations.

A09 - Security Logging and Monitoring Failures

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.

A10 - Server-Side Request Forgery (SSRF)

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

“ fabien.kleinbourg+opsie[at]gmail[dot]com
f.kleinbourg[at]univ-lyon2[dot]fr ”