Introduction to the OWASP Top 10 – 2021

Risk A10: Server-Side Request Forgery

Key Concepts

Key Concept 1: Server-Side

- Programs and operations that run on the server, like an API on the backend
- Contrast to processes that run on the client, like a browser or mobile app

Key Concept 2: Request Forgery

- When an attacker tricks an entity into making unintended, forged requests
- Request Forgery is about forcing an entity into making a request that they never intended to make

Key Concept 3: Cross-Site Request Forgery

- An attack that forces the user to execute unwanted actions while authenticated to a target web application

Additional Information

Server-side refers to programs and operations that run on the server, like an API on the backend. It is the opposite to processes that run on the client, like a browser or mobile app.

Request forgery is forcing a user or other entity to make a forged web request that they never intended to make. Simply put, request forgery is about tricking a user or other entity into making a request to a web application or API with malicious data – one they never intended to make.

Cross-Site Request Forgery is an attack that forces the user to execute an unwanted action, while they are authenticated to your web application. Think of the user receiving a phishing e-mail with a URL authorizing a transaction on their web banking account and clicking on it, while logged in to their bank account website. If they click on that link, while logged in to their web banking account, they have fallen victim to a Cross-Site Request Forgery attack.

A one-click attack involves sending a malicious URL to an authenticated user, that performs an action they do not approve or know. One-click attacks are a specific type of Cross-Site Request Forgery attacks.

Definition

Server-Side Request Forgery (or SSRF) occurs when an attacker sends malicious input to a server, forcing the server to make a request to an unintended resource.

- Can lead to a variety of impacts involving loss of data, privilege escalation and more
- Server-side request forgery is a common vulnerability in the world of micro-services or N-tiered webservices

Additional Information

The definition of A10 of 2021 – Server-Side Request Forgery – is forcing a server to make a request to an unintended resource.

A10 can lead to a variety of impacts involving loss of data, privilege escalation and more. Server-Side Request Forgery is a common vulnerability in the world of microservices or multi-tiered web services. We often refer to such applications as microservices. Thus, Server-Side Request Forgery vulnerabilities are a common risk within microservices.

Example

To demonstrate our good and bad example, consider our application receiving a URL from the user that is added (in some form) to the existing page being displayed.

Bad Example

```
addToPage(String url) {
 if (Validator.isValidURL(url)) {
    return fetchContent(url);
 }
}
```

Good Example

```
addToPage(String url) {
 if (Validator.isValidURL(url)) {
   if (url.domain == "manicode.com") {
      return fetchContent(url);
   }
  }
}
```

Additional Information

For our bad example, the URL, which is user defined input is only checked for validity and nothing more.

For our good example, the URL is also checked to come from a trusted domain. Only if the URL is valid and comes from a trusted domain do we add it to the page.

Challenges

Traditional Code and Dynamic Scanning Tools Struggle

Tools struggle to accurately identify Server-Side Request Forgery. As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario

Passing URLs and IP addresses is quite common for logging purposes

- Typical cloud metadata storage over HTTP on a specific URL (http://169.254.169.254/)
- “Internal” services such as a service listening on localhost only (http://localhost:28017/)

Additional Information

We now understand how the risk of Server-Side Request Forgery can materialize, let us look at why this is a common issue.

Traditional code and dynamic scanning tools struggle to accurately identify Server-Side Request Forgery. As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario.

Passing URLs and IP addresses is quite common for example, for logging purposes. Think of your typical cloud metadata storage over HTTP on a specific URL (http://169.254.169.254/) or think of “internal” services such as a service listening on localhost only (http://localhost:28017/).

This combination of passing URLS and IP addresses, coupled with the fact that *Server-Side Request Forgery is hard to accurately detect*, creates a breeding space for this risk.

Best Protection Strategies

VEST

Validate origin of URLs and IPs when parameters

Ensure authentication and access control on APIs

Setup URL encoding for untrusted parameters

Test and limit network service access with network controls

Additional Information

Above you have some of the best protection strategies against A10 of the Top 10 – Server-Side Request Forgery.