web security

Web□□□□□□□□□□□□□□□□□□□□

1. **Html□□□□□□□**
   1. □□□□□□□Javascript□□□□□□□□□□□□□□□□CSP□□□□□□□□□□
   2. **□□ iframe□□□clickjacking□□□**iframe□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
2. **□□□Injection□□**□□□□□□□text□□□□□□□□□□□□□□□□□□□JS/PHP/SQL/Bash □□□ text □□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□SQL□□□□□□□□□□XSS□Cross Site Script□□□□□□□□□□□&CSP
3. **HTTP□□□□**□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□Web Security□□□□□□□□ik2p4Rz4QzM□□□□HTTP□□□□BurpSuite□□□□□□□DVWA□□□□□□SQL/Bash/XSS□□□□□□□□□□□□□□□□□□Host□□□Brute Force□□□□□□

CSP(content-security-policy)

> The basic idea behind CSP is to block inline script execution and provide the list of allowed sources of trusted content (scripts, stylesheets, fonts, plugins, etc.) to the browser. Even if an attacker gets to inject their bad script, the browser won't execute it since CSP prevents inline script execution.

□□□□□□□□□□□□□□□□□

```
// http res header
Content-Security-Policy: script-src 'self' https://safe-external-site.com; style-src 'self'

// or
<meta http-equiv="Content-Security-Policy" content="script-src 'self' https://safe-ext
```

□□ **CSP**

```
Content-Security-Policy: script-src 'self' 'unsafe-inline' https://safe-external-site.com
```

□□□□ inline script □□

> It tells the browser that these elements were not injected by the hacker (since they couldn't guess the nonce value), but were intentionally inserted by the server, so they're safe to execute.

```
<script nonce="dGhpcyBpcyBhIG5v==">.. ..</script>
<style nonce="dGhpcyBpcyBhIG5v==">.. ..</style>

Content-Security-Policy: script-src 'nonce-dGhpcyBpcyBhIG5v=='; style-src 'nonce-dG
```

□□□□□□□

TSL □□[1]

> - □□□□□□□□□□□□□□□□□□□□
> - □□□□□ `Cert□□` □□□□
> - □□□□□□□□□□□
> - □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Cert

Certificate Authority

XXXXXXXXXXXXXXX
X OS X
X X

```
X  +---------+ X +----+   +-----------+

X  | Root CA +-->+ CA +-->+ 'Leaf' CA |
X  +---------+ X +----+   +-----------+
X           X
X    pubkey   X  pubkey
X          Sign
X    prikey+---->Cert
X     +     X        Sign
X  Sign|     X  prikey+--->Cert
X     v     X
X    Cert    X
XXXXXXXXXXXXXXXX

message             Sign
+ ^         Issuer+--->Subject
encrypt\w| |decrypt\w
prikey   | |pubkey    prikey+--->Cert=Signature+Issuer.pubkey
v +
encryption      CertB.hash=Hash(CertA.pubkey+CertB.Signature)
```

clickjacking-attacks

Desc

- □□□□□□□□□□□□□□□□□□□vulnerable_website□A
- □□□□□□□□□attacker_website□B
- □ A □□□ B□□ B □□□ A □ iframe□□□ A □ B □□□□□□□ A □ z-index □□ B□
- □□□□□□□□□ B □□□□□□□□□□□□□□□ A □□□□□□□□□□□□

□□[2] [2]" role="complementary" aria-hidden="true">#

Client-side defenses

frame-busting

```
<html>
<head>
<title>Vulnerable Page</title>
<script>
if (top != window) {
/**
* □□□□□□□□□□ 100% □□
* □□□□□□ window.onbeforeunload event, □□ Attracker Page □□□□□□□
* window.onbeforeunload = () => false;
* □□□□□□□□□□
*/
top.location = window.location;
}
</script>
</head>
</html>
```

```
<html>
<head>
<title>Attracker Page</title>
<script>
window.onbeforeunload = function () {
return false;
};
</script>
</head>
<body>
<iframe id="vulnerable_website" src="http://Vulnerable.Page" sandbox="allow-scrip
</iframe>
</body>
</html>
```

□□□□□client-side □□ block clickjacking attacks □□□□□□

□□ X-Frame-Options header

```
// server.js
app.use(function (req, res, next) {
res.setHeader("X-Frame-Options", "sameorigin");
next();
});
```

□□□□□□□□□□ X-Frame-Options□□□□

Using CSP / Content-Security-Policy

```
// server.js
app.use(function (req, res, next) {
// same as { X-Frame-Options: "sameorigin" }
res.setHeader("Content-Security-Policy", "frame-ancestors 'self';");

// frame-ancestors 'none' - Not allowed to use frame at all.
// frame-ancestors https://www.authorized-website.com - Only allowed at specific w
next();
});
```

□□□X-Frame-Options □□□□

Using cookie's sameSite origin

```
// server.js
app.use(
session({
secret: "my-secret",
resave: true,
saveUninitialized: true,
cookie: {
httpOnly: true, // Cookie □□□□□□□□□□□Js □□□□□□
sameSite: "strict", //□ new code
},
})
);
```

XSS

> XSS is one of the most common security vulnerabilities. An attacker can try to inject this script anywhere they can submit a form

1. □□□□[4]
2. CSP

css hajack

□□□□□□□□[3]

```
input[value=a] { backgroud: url(https://example.com/?value=a)}
```

---

1. https://cjting.me/2021/03/02/how-to-validate-tls-certificate/ □
2. https://auth0.com/blog/preventing-clickjacking-attacks/ □
3. https://scotthelme.co.uk/can-you-get-pwned-with-css □
4. https://www.writesoftwarewell.com/content-security-policy/ □

Date: 2024-04-04
Words: 1124
Time to read: 5 mins

Newer Older

2024-04-05
[JavaScript] clone

2024-03-25
bundle