CORS

Created by @mehdi0x90

ACAO

```
GET /sensitive-victim-data HTTP/1.1
Host: vulnerable-website.com
Origin: https://malicious-website.com
Cookie: sessionid=...
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://malicious-website.com
Access-Control-Allow-Credentials: true
...
```

Errors parsing Origin headers

- grants access to all domains ending in normal-website.com
- attacker might be able to gain access by registering the domain hackersnormal-website.com

Whitelisted null origin value

```
GET /sensitive-victim-data
Host: vulnerable-website.com
Origin: null
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: null
Access-Control-Allow-Credentials: true
```

- Some applications might whitelist the null origin to support local development of the application
- Browsers might send the value null in the Origin header in various unusual situations
  - Cross-origin redirects
  - Requests from serialized data
  - Request using the file: protocol
  - Sandboxed cross-origin requests
- For example, this can be done using a sandboxed iframe cross-origin request of the form

Exploiting XSS via CORS trust relationships

Breaking TLS with poorly configured CORS

Intranets and CORS without credentials

```
GET /reader?url=doc1.pdf
Host: intranet.normal-website.com
Origin: https://normal-website.com
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

The application server is trusting resource requests from any origin without credentials. If users within the private IP address space access the public internet then a CORS-based attack can be performed from the external site that uses the victim's browser as a proxy for accessing intranet resources.