Supercookies

In this lesson, we'll study supercookies.

We'll cover the following

Introduction

What if we were able to set a cookie on a top-level domain (TLD) such as `.com` or `.org`? That would be a huge security concern, for two reasons:

- **user privacy**: every website running on that specific TLD would be able to track information about the user in shared storage
- **information leakage**: a server could mistakenly store a sensitive piece of data in a cookie available to other sites

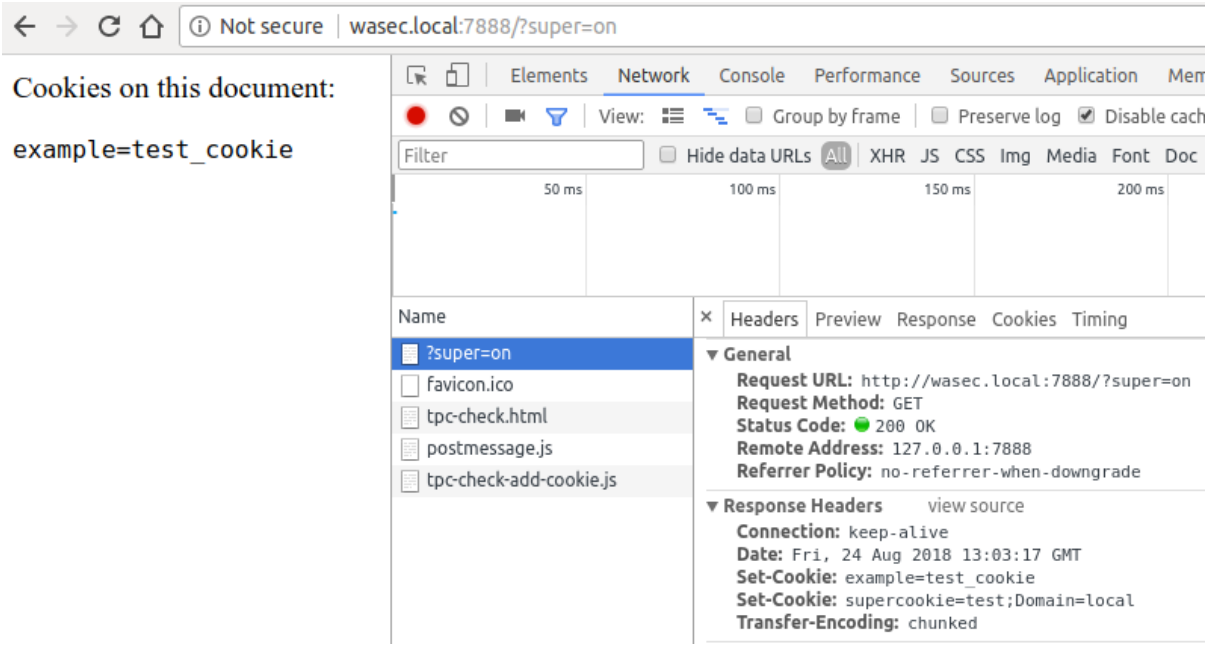
In the following code, by appending `?super=on` to the URL, the server is going to set a cookie on the domain local (e.g., wasec.local). Since it is a top-level domain, the client will refuse to process this cookie.

```
var qs = require('querystring')
var url = require('url')
var fs = require('fs')

function server (req, res) {
  if (req.url === '/same-site-form') {
    return res.end(`
<html>
  <form action="http://wasec.local:7888/" method="POST">
    <input type="hidden" name="destination" value="attacker@email.com" />
    <input type="hidden" name="amount" value="1000" />
    <input type="submit" value="CLICK HERE TO WIN A HUMMER" />
  </form>
</html>
`)
  }
  let query = qs.parse(url.parse(req.url).query)
```

```
  if (query.clear === "on") {
    res.writeHead('302', {
      'Set-Cookie': [

        'example=a;Expires=Wed, 21 Oct 2015 07:28:00 GMT',
        `example_with_domain=a;Domain=wasec.local;Expires=Wed, 21 Oct 2015 07:28:00 GMT`,
        `supercookie=a;Expires=Wed, 21 Oct 2015 07:28:00 GMT`,
        `secure=a;Expires=Wed, 21 Oct 2015 07:28:00 GMT`,
        `not_secure=a;Expires=Wed, 21 Oct 2015 07:28:00 GMT`,
      ],
      'Location': '/'
    })
    res.end()
    return
  }

  let headers = {}
  headers['Set-Cookie'] = []

  if (!req.headers.host.startsWith('sub.wasec.local')) {
    headers['Set-Cookie'].push('example=test')

    if (query.domain === 'on') {
      headers['Set-Cookie'].push(`example_with_domain=test_domain_cookie;Domain=wasec.local`)
    }
  }

  if (query.super === 'on') {
    headers['Set-Cookie'].push(`supercookie=test;Domain=local`)
  }

  if (query.secure === 'on') {
    headers['Set-Cookie'].push(`secure=test;Secure`)
  }

  if (query.secure === 'on') {
    headers['Set-Cookie'].push(`not_secure=test`)
  }

  if (query.httponly === 'on') {
    headers['Set-Cookie'].push(`http_only_cookie=test;HttpOnly`)
  }

  if (query.samesite === 'on') {
    headers['Set-Cookie'].push(`same_site_cookie=test;SameSite=Lax`)
  }

  res.writeHead(200, headers)
  res.end(`
<html>
  <div id="output"/ >
  <script>
    let content = "none";
    if (document.cookie) {
      let cookies = document.cookie.split(';')
      content = ''
      cookies.forEach(c => {
        content += "<p><code>" + c + "</code></p>"
      })
    }
    document.getElementById('output').innerHTML = "Cookies on this document: <div>" + content + "</
  </script>
<html>
```

```
`)
}

const options = {
  key: fs.readFileSync(__dirname + '/../wasec.local-key.pem'),
  cert: fs.readFileSync(__dirname + '/../wasec.local.pem')
};

require('http').createServer(server).listen(7888)
require('https').createServer(options, server).listen(7889)
```

Luckily, TLD-cookies, otherwise known as supercookies, are disabled by web browsers for the reasons I mentioned above. If you try to set a supercookie, the browser will simply refuse to do so. If we append the parameter `super=on` in our example, we will see the server trying to set a supercookie, while the browser ignores it. Try it with **https://x6jr4kg.educative.run/?super=on** but replace this URL with the one generated on your app above.



Supsercookie attempted to be set by server

In today's web, there are other ways to keep track of users, ETag tracking being one example. Since cookies are usually associated with tracking, these techniques are often referred to as supercookies as well, even though they do not rely on HTTP cookies. Other terms that may refer to the same set of technologies and practices are *permacookies* (permanent cookies) or *zombiecookies* (cookies that never die).

Unwanted Verizon ads

Companies love to make money out of ads, that's no news. But when ISPs start to aggressively track their customers in order to serve unwanted ads, well

> to aggressively track their customers in order to serve unwanted ads, well, that's a different story.
>
> In 2016, Verizon was found guilty of tracking users without their consent, and sharing their information with advertisers. This resulted in a fine of $1.35 million and the inability, for the company, to continue with their questionable tracking policy.
>
> Another interesting example was Comcast, who used to include unwanted ads through custom JavaScript code in web pages served through its network.
>
> Needless to say, if all web traffic would be served through HTTPS we wouldn't have this problem as ISPs wouldn't be able to decrypt and manipulate traffic on-the-fly.

Cookie flags that matter

We've barely scratched the surface of HTTP cookies, it's now time for us to dig even deeper.

There are three very important directives (`Secure`, `HttpOnly`, and `SameSite`) that should be understood before using cookies as they heavily impact how cookies are stored and secured. Let's study them in the next lesson!