

A large muraena eel is shown swimming over a sandy ocean floor. The eel's body is covered in dark, mottled patterns, and it has a long, pointed snout. It is swimming towards the right side of the frame. The background consists of green seagrass and sandy ocean floor.

MURAENA

antisnatchor & ohpe present

THE
UNEXPECTED
PHISH

OUTLINE

- ▶ Demystifying 2FA
- ▶ Reverse Proxy strikes back
 - ▶ Reversing anti-reverse proxy
- ▶ The art of instrumenting stolen web sessions
 - ▶ Bypassing anti-browser instrumentation
 - ▶ Automating post-phishing activities
- ▶ Full-chain demos on GSuite, GitHub, Dropbox
- ▶ Upcoming challenges

DEMYSTIFYING 2FA: NON-U2F MYTHS

- ▶ You would immediately realise something dodgy is going on if you receive a 2FA challenge/token for an action you did not perform
- ▶ If attackers know your credentials, they can't login
- ▶ If you set up 2FA you are more secure (even via SMS)

DEMYSTIFYING 2FA: NON-U2F MYTHS

- ▶ You would immediately realise something dodgy is going on if you receive a 2FA challenge/token for an action you did not perform
- ▶ Potentially true, but it doesn't consider scenarios where both credentials and tokens are phished in real-time
- ▶ If attackers know your credentials, they can't login
- ▶ It minimises only password spraying and other dictionary related attacks (Captcha would be enough)

DEMYSTIFYING 2FA

- *If you set up 2FA you are more secure (even via SMS)*

MitiGator raises the bar...



...until it sees no more exploits

Credit @halvarflake

DEMYSTIFYING 2FA: NON-U2F MYTHS

- ▶ Universal Two Factor (U2F) is the only 2FA solution that offers protection from phishing
 - ▶ Crypto challenge with the web origin
- ▶ Bypassed in early 2018 via WebUSB by @marver and @antisnatchor
 - ▶ OffensiveCon talk at: youtube.com/watch?v=pUa6nWWTO4o

REVERSE PROXY TO THE RESCUE

- ▶ Most 2FA solutions rely on token submission **via a web form**: SMS, Push/OTP based, Google/CompanyX authenticators
- ▶ After credentials are verified, the server triggers the 2FA action and waits for user token submission
- ▶ A smart **reverse proxy** can then be used to:
 - ▶ **intercept** all the traffic
 - ▶ **fulfil** requests that trigger the 2FA action
 - ▶ **pass** post-2FA login session cookies to an instrumented browser that **hijacks** the victim's session

REVERSE PROXY TO THE RESCUE

- ▶ There is no magic here (and don't say you didn't know):



REVERSE PROXY TO THE RESCUE

- ▶ Reverse proxies exist from over 20 years
- ▶ Technical Trends in Phishing Attacks (US CERT) - 2005

https://www.us-cert.gov/sites/default/files/publications/phishing_trends0511.pdf

3.4.4 Man-in-the-Middle Attacks

Man-in-the-middle attacks define a broad class of potential attacks in which an attacker is able to intercept, read, and modify communications between two other parties without their knowledge. As related to phishing, a man-in-the-middle attack involves an attacker serving as a proxy between a user and an online commerce site. The attacker potentially has access to all authentication and account information, including an opportunity to hijack credentials used in two-factor authentication.

- ▶ So how did the situation evolved from 2005?

REVERSE PROXY TO THE RESCUE

- ▶ In the last 4/5 years they have been hot again and used for offensive purposes:
 - ▶ (2015) <https://labs.portcullis.co.uk/blog/blood-in-the-water-phishing-with-beef/>
 - ▶ (2017) <http://www.chokepoint.net/2017/03/reverse-proxy-phishing-with-valid.html>
 - ▶ (2017) <https://breakdev.org/evilginx-advanced-phishing-with-two-factor-authentication-bypass/>

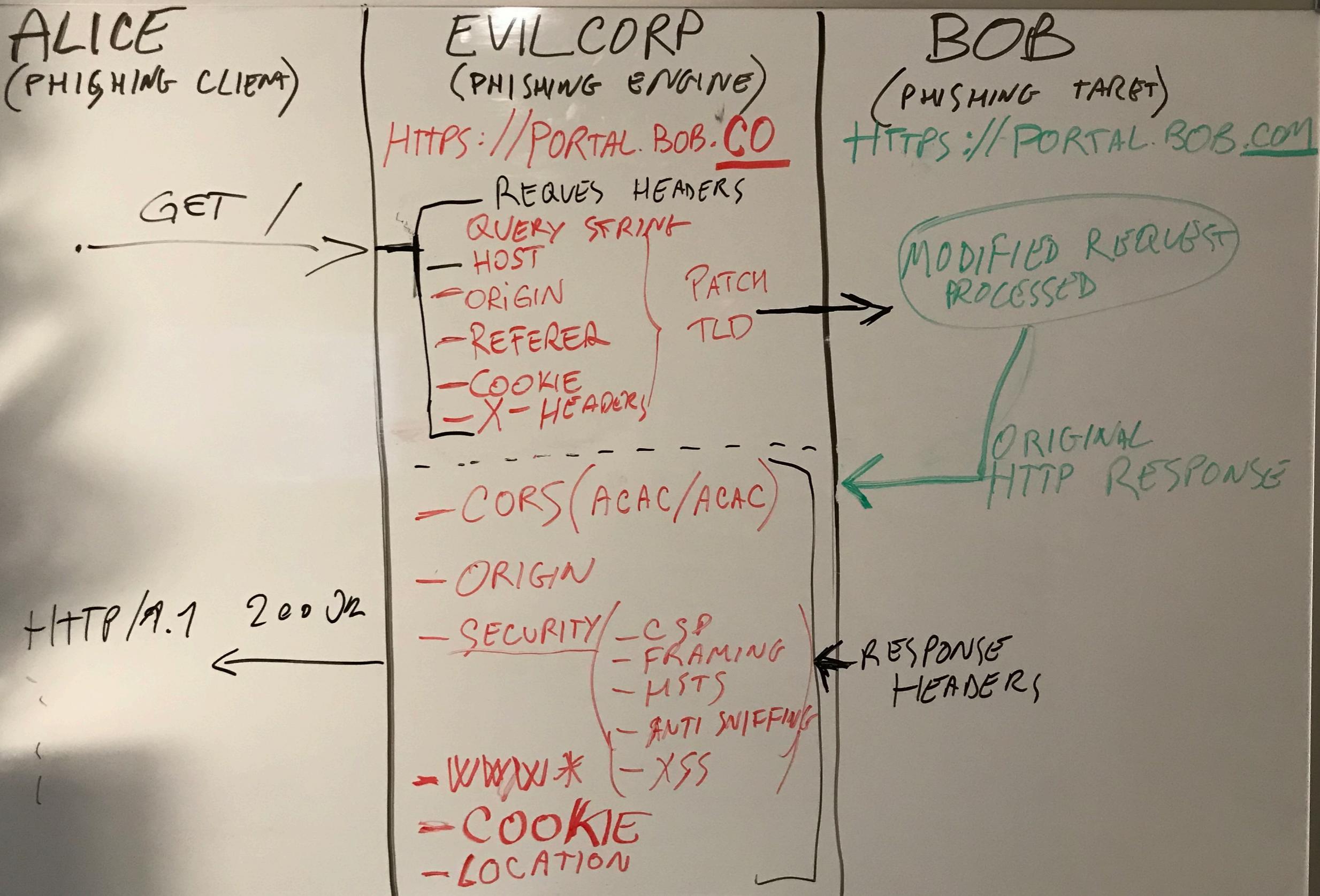
REVERSE PROXY TO THE RESCUE

- ▶ And as expected on APTs:
 - ▶ (2017) <https://citizenlab.ca/2017/02/nilephish-report/>
 - ▶ (2018) <https://www.amnesty.org/en/latest/research/2018/12/when-best-practice-is-not-good-enough/>



- ▶ How did the community react to this?

REVERSE PROXYING HEADERS



REVERSE PROXYING BODY

ALICE
(PHISHING CLIENT)

EVIL CORP
(PHISHING ENGINE)

HTTPS://PORTAL.BOB.CO

GET /

HTTP REQUEST BODY

= MATCH/REPLACE
FQDN

= UPDATE CONTENT LENGTH

HTTP RESPONSE BODY

= DECOMPRESSOR
(GZIP
DEFLATE
BR)

= CONTENT-TYPE

WHITELIST

PREVENTS
PROXYING
VIDEO/AUDIO
IMAGE/CSS
BINARY

= MATCH/REPLACE FQDN

HTTP/1.1 200 OK

BOB
(PHISHING TARGET)

HTTPS://PORTAL.BOB.COM

MODIFIED REQUEST
PROCESSED

ORIGINAL
HTTP RESPONSE

STATIC REPLACE

RULE-BASED REPLACE

- DEOBfuscate/MODify
PACKED JAVAScript

- CHANGE STRINGS
VIA REGEXES

REVERSE PROXY: MURAENA GOLANG IMPLEMENTATION

- ▶ We choose Golang for:
 - ▶ High performance, great syntax
 - ▶ Stable core and good library ecosystem
 - ▶ Cross-compilation
- ▶ `cloc . --exclude-dir vendor:` ~2300 LoC



REVERSE PROXY: MURAENA GOLANG IMPLEMENTATION

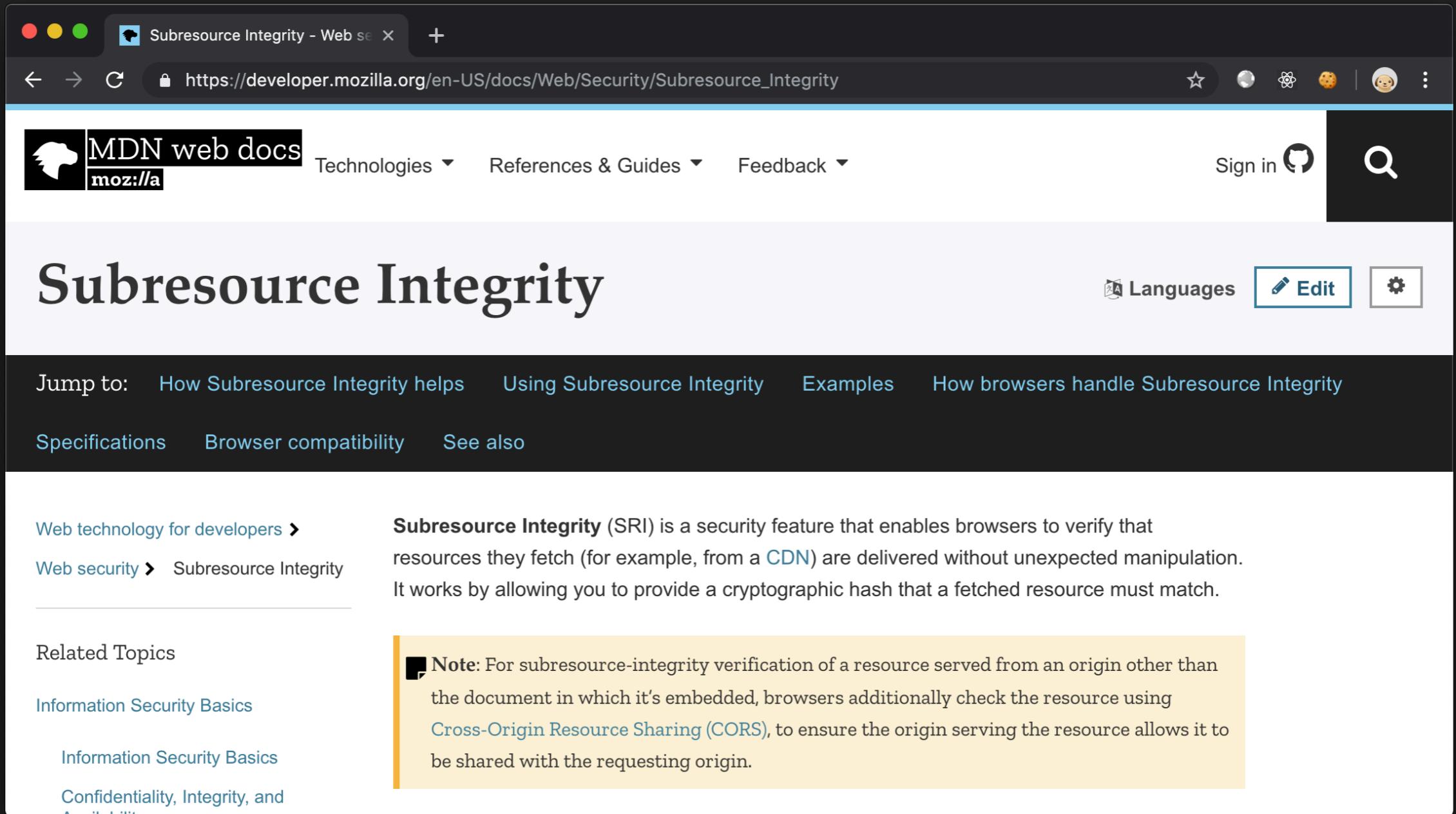
- ▶ Muraena comes with a number of unique features
 - ▶ Web Crawler
 - ▶ Customisable Tracking
 - ▶ Static Content Server
 - ▶ Wildcard domain support
 - ▶ Browser Instrumentation Integration
- ▶ The proxy core is `net/http SingleHostReverseProxy` with a custom transformer to grep & replace strings in HTTP request/response

REVERSING ANTI-REVERSE PROXY

- ▶ Some origins implement additional checks aimed at preventing proxying or framing
- ▶ Mostly easy checks, but reversing could get complicated with polymorphic heavily-obfuscated Javascript
- ▶ Chrome DevTools and Burp Proxy are your friends ;-)

REVERSING ANTI-REVERSE PROXY

► Universal manipulations



The screenshot shows a macOS desktop environment with a dark-themed browser window. The title bar reads "Subresource Integrity - Web se X". The address bar shows the URL "https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity". The page content is from MDN web docs, featuring a large title "Subresource Integrity". Below the title are navigation links: "Jump to: How Subresource Integrity helps Using Subresource Integrity Examples How browsers handle Subresource Integrity", "Specifications", "Browser compatibility", and "See also". On the left sidebar, there are links to "Web technology for developers >" and "Web security > Subresource Integrity". A "Related Topics" section lists "Information Security Basics" and "Confidentiality, Integrity, and Availability". The main content area describes Subresource Integrity (SRI) as a security feature that verifies resources from a specific origin. A note box states: "Note: For subresource-integrity verification of a resource served from an origin other than the document in which it's embedded, browsers additionally check the resource using Cross-Origin Resource Sharing (CORS), to ensure the origin serving the resource allows it to be shared with the requesting origin." The MDN logo and navigation links like "Technologies", "References & Guides", "Feedback", "Sign in", and "Languages" are visible at the top of the page.

REVERSING ANTI-REVERSE PROXY

► Universal manipulations

```
<script src="https://example.com/example-framework.js"
    integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/
uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
    crossorigin="anonymous"></script>
```

becomes

```
<script src="https://example.com/example-framework.js"
    no-integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K
uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
    crossorigin="anonymous"></script>
```

REVERSING ANTI-REVERSE PROXY

- ▶ Universal manipulations
- ▶ Content-Security Policy annihilation!

```
<script src="https://example.com/example-framework.js"  
nonce="oqVuAfXRKap7fdgc"></script>
```

becomes

```
<script src="https://example.com/example-framework.js"  
no-more-nonce="oqVuAfXRKap7fdgc"></script>
```

REVERSING ANTI-REVERSE PROXY

- ▶ Universal manipulations
- ▶ Content-Security Policy annihilation!

<meta http-equiv="Content-Security-Policy" content="...">>

becomes

<meta no-more-CSP content="...">>

REVERSING ANTI-REVERSE PROXY

- ▶ Universal manipulations
- ▶ Content-Security Policy annihilation!

```
"remove": {  
    ...,  
    "response": {  
        "header": [  
            "Content-Security-Policy",  
            "Content-Security-Policy-Report-Only",  
            ...  
        ]  
    }  
},
```

REVERSING ANTI-REVERSE PROXY: GITHUB



► An easy one: GitHub

The screenshot shows a browser window with the URL `view-source:https://github.com`. The page content displays several `<meta>` tags. One tag at line 98 contains a long, encoded string:

```
content="MDI1YzE2ZmE0YzRlMDlmZmY4YzZmNT4MKRF0jM1NkFGMjk6NUNEMTk50DMiLCJ0aW1lc3RhbXAiOjE1NTcyNDAxOTUsImhvc3QiOiJnaXRodWIuY29tIn0="
```

Line 100 shows another tag with a long content attribute containing a JSON object:

```
content="UNIVERSE_BANNER,MARKETPLACE_INVOICED_BILLING,MARKETPLACE_SOCIAL_PROOF_CUSTOMERS,MARKETPLACE_TRENDING_SOCIAL_PROOF,MARKETPLACE_RECOMMENDATIONS">
```

Line 102 shows a tag with a nonce attribute:

```
<meta name="html-safe-nonce" content="37f72641ec81fb48d10f5a10bb226d9024a84c4f">
```

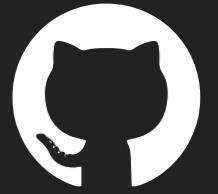
Below the page content, the browser's developer tools are visible, specifically the Console tab. It shows two log entries. The first entry is a call to `atob` with the same long string from the meta tag:

```
> atob('MDI1YzE2ZmE0YzRlMDlmZmY4YzZmNT4MKRF0jM1NkFGMjk6NUNEMTk50DMiLCJ0aW1lc3RhbXAiOjE1NTcyNDAxOTUsImhvc3QiOiJnaXRodWIuY29tIn0=')
```

The second entry is a response object:

```
< "025c16fa4c4e09fff8c6f521104c234c76b31ed587605cb492524bcbbc24105e|{"remote_address":"[REDACTED]","request_id":"F428:7691:1FC82DE:356AF29:5CD19983","timestamp":1557240195,"host":"github.com"}"
```

REVERSING ANTI-REVERSE PROXY: GITHUB



► An easy one: GitHub

The screenshot shows a browser window with multiple tabs. The active tab is "view-source:https://github.com". The main content area displays the GitHub homepage. A debugger is active, with the status bar showing "Paused in debugger". The code being analyzed is from the file "github-bootstrap...9.js:formatted". The specific line highlighted is:

```
const qc = document.querySelector("meta[name=js-proxy-site-detection-payload]");
, Cc = document.querySelector("meta[name=expected-hostname]");
if (qc instanceof HTMLMetaElement && Cc instanceof HTMLMetaElement && vt(document)) {
  const e = {
    url: window.location.href,
    expectedHostname: Cc.content,
    documentHostname: document.location.hostname,
    proxyPayload: qc.content
  }
  , t = new Error
  , n = {};
  n.$__ = btoa(JSON.stringify(e)),
  ze(t, n)
}
```

The code is part of a conditional block that checks if the document contains specific meta tags related to proxy detection. The debugger interface includes a sidebar with various developer tool options like Elements, Network, and Performance.

REVERSING ANTI-REVERSE PROXY: GITHUB



- ▶ An easy one: GitHub
- ▶ The fix: nullify the keywords:

"js-proxy-site-detection-payload" → ""

"expected-hostname" → ""

REVERSING ANTI-REVERSE PROXY: DROPBOX



► The reCAPTCHA challenge

The screenshot shows a web browser window for Dropbox at <https://www.dropbox.com>. The main content area features a large, bold headline: "Put your creative energy to work with Dropbox". Below the headline is a subtext: "Dropbox is a modern workspace designed to reduce busywork – so you can focus on the things that matter." To the right is the sign-in form. It includes fields for "Email" and "Password", a "Remember me" checkbox, a blue "Sign in" button, and a "Sign in with Google" button. A note at the bottom of the form states: "This page is protected by reCAPTCHA and is subject to the Google [Privacy Policy](#) and [Terms of Service](#)". This note is highlighted with a red rectangular box. At the bottom right of the sign-in form is a link: "Forgotten your password?".

Dropbox

For teams For individuals

Sign up Download

Put your creative energy to work with Dropbox

Dropbox is a modern workspace designed to reduce busywork – so you can focus on the things that matter.

Sign in
or [create an account](#)

Email

Password

This page is protected by reCAPTCHA and is subject to the Google [Privacy Policy](#) and [Terms of Service](#).

Remember me

Sign in

Sign in with Google

[Forgotten your password?](#)

REVERSING ANTI-REVERSE PROXY: DROPBOX



► The reCAPTCHA challenge

The screenshot shows a browser window with two tabs. The active tab is titled "Dropbox" and displays a phishing page for "www.phishing.anti/#". The page has a dark red background with white text: "Put your creative energy to work with Dropbox". At the top right are "Sign up" and "Download" buttons. Below the main text is a reCAPTCHA form with fields for "Email" and "Password". A red error message states: "This page is protected by reCAPTCHA and is subject to the Google Privacy Policy and Terms of Service." To the right of the error message is a reCAPTCHA logo and links for "Privacy - Terms". The bottom of the page features a standard footer with links like "About", "Help", "Support", "Blog", "Press", "API", "Jobs", "Terms of Service", and "Privacy Policy".

The browser's developer tools are open, specifically the "Elements" tab in the bottom-left corner. The "Elements" panel shows the DOM structure of the page, with a script tag containing a reCAPTCHA initialization function. The function code is as follows:

```
recaptcha.anchor.ErrorMain.init("[\x22input\x22,null,null,null,null,null,[1,1,1]\n,\x22Invalid domain for site key\x22,6,null,null,null,[\x22https://drb-17.phishing.anti/intl/en-GB/policies/privacy/\x22,\x22https://drb-17.phishing.anti/intl/en-GB/policies/terms/\x22]\n]\n");
```

The "Styles" panel on the right shows that no matching selector or style was found for the current context.

REVERSING ANTI-REVERSE PROXY: DROPBOX



► The reCAPTCHA challenge

The screenshot shows a browser window with three tabs, all titled "Dropbox". The middle tab is active and displays a phishing page at <https://www.phishing.anti/>. The page has a red header with the Dropbox logo and links for "For teams" and "For individuals". Below the header is a cookie consent message: "We use cookies so that Dropbox works for you. By using our website, you agree to our use of cookies. [Learn more](#)".

The browser's developer tools Network tab is open, showing requests for "/recaptcha". One request is selected, showing details:

- Name:** recaptcha_v2_challenge-v15559686297
- Type:** CSS
- Request URL:** <https://drb-17.phishing.anti/recaptcha/api2/anchor?ar=1&k=6LdnLyIUAAAAA0iGPtdh-g3KiJRoDGGPD-6dqXo&co=aHR0cHM6Ly93d3cucGhpc2hpbmcuYW50aTo0NDM.&hl=en-GB&v=v15559686297>
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** 127.0.0.1:31337
- Referrer Policy:** origin-when-cross-origin

The response headers include:

- Alt-Svc: `quic=:443"; ma=2592000; v="46,44,43,39"`
- Cache-Control: private, max-age=0
- Connection: close

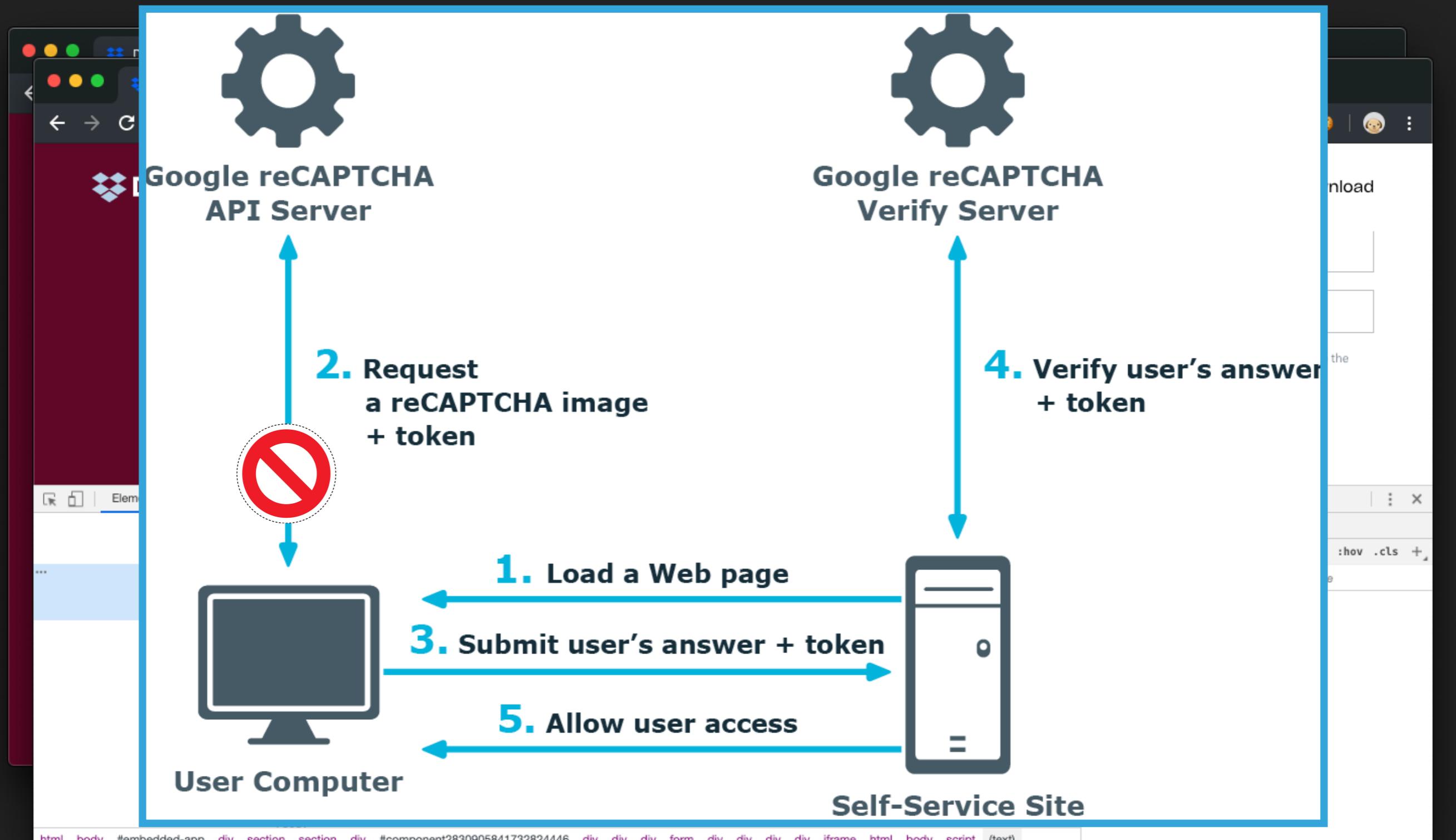
The bottom of the developer tools shows a console with the following entries:

```
atob('aHR0cHM6Ly93d3cucGhpc2hpbmcuYW50aTo0NDM')
"https://www.phishing.anti:443"
>
```

REVERSING ANTI-REVERSE PROXY: DROPBOX



► The reCAPTCHA challenge



REVERSING ANTI-REVERSE PROXY: DROPBOX



- ▶ The reCAPTCHA challenge
- ▶ The fix: base64 transformation support:

```
"transform": {  
    "base64": {  
        "enabled": true,  
        "padding": [  
            "=",  
            "..."  
        ]  
    },
```

REVERSING ANTI-REVERSE PROXY: GSUITE



- ▶ If it works for Google works for all



REVERSING ANTI-REVERSE PROXY: GSUITE



- ▶ If it works for Google works for all
- ▶ There are several regexes to patch:

```
["| (google))\\.com","|(google)|(phishing)(\\.anti|\\.com")",
[".google\\.((co|com)", ".(google|phishing)\\.(((co|com|anti)",
[".google\\.com", ".phishing\\.anti"],
["\\.google(rs)?\\.com", "\\.(google(rs)|phishing)?\\.(com|anti)",
["LCJwcHUi0iJodHRwczovL21haWwuZ29vZ2x1LmNvbS9yb2JvdHMudHh0IiwibHB1IjoiaHR0cHM6Ly
9oYW5nb3V0cy5nb29nbGUuY29tL3JvYm90cy50eHQifQ",
"LCJwcHUi0iJodHRwczovL21haWwucGhpc2hpbmcuYW50aS9yb2JvdHMudHh0IiwibHB1IjoiaHR0cHM
6Ly9oYW5nb3V0cy5waGlzaGluZy5hbnRpL3JvYm90cy50eHQifQ=="
],
["LCJwcHUi0iJodHRwczovL2hhbmdvdXRzLmdvb2dsZS5jb20vcn9ib3RzLnR4dCIsImxwdSI6Imh0dH
Bz0i8vaGFuZ291dHMuZ29vZ2x1LmNvbS9yb2JvdHMudHh0In0",
"LCJwcHUi0iJodHRwczovL2hhbmdvdXRzLnBoaXNoaw5nLmFudGkvcm9ib3RzLnR4dCIsImxwdSI6Imh
0dHBz0i8vaGFuZ291dHMucGhpc2hpbmcuYW50aS9yb2JvdHMudHh0In0="]
```

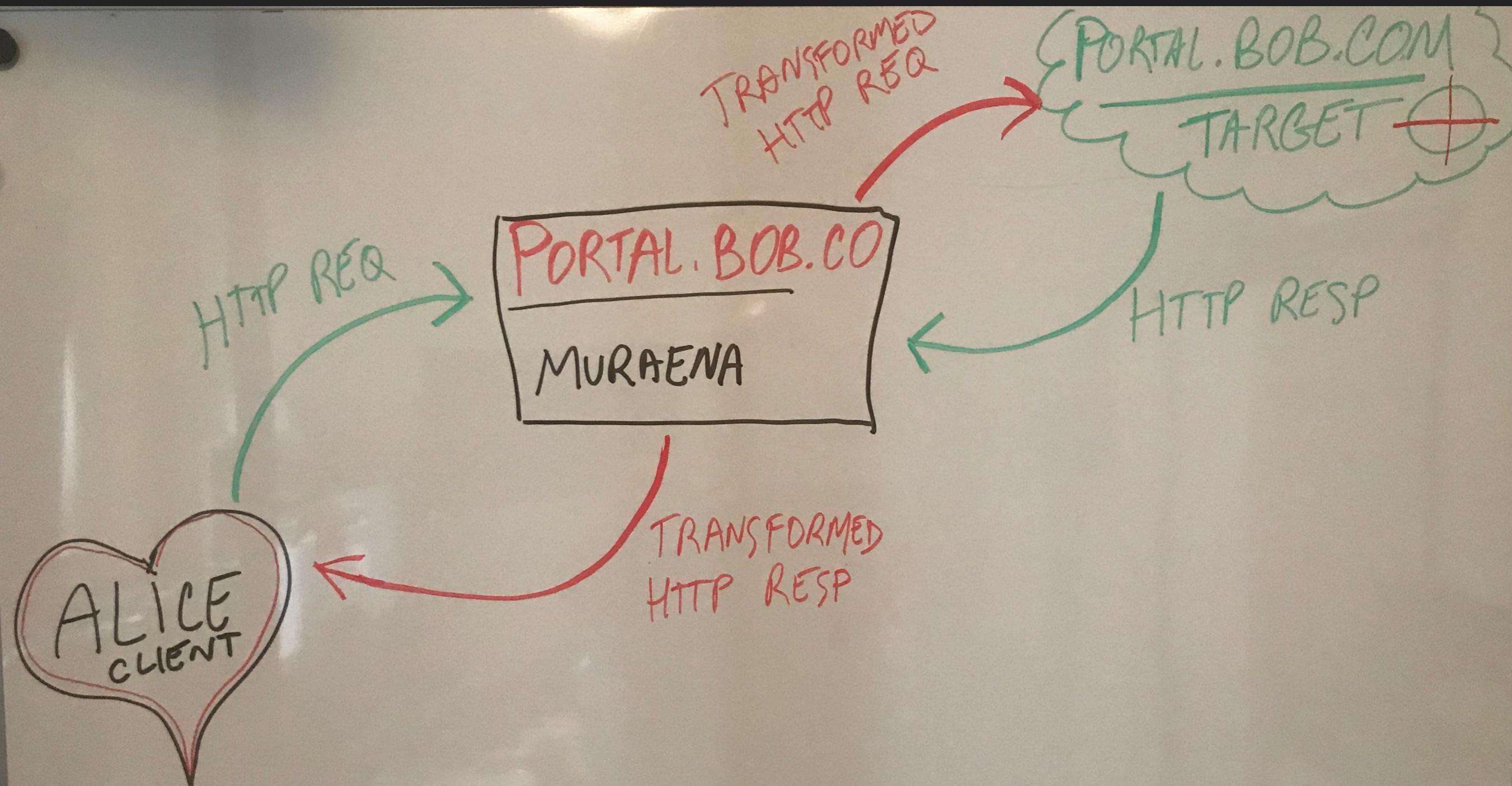
THE ART OF INSTRUMENTING STOLEN WEB SESSIONS

- ▶ Since all the traffic is passing through Muraena, credentials and session cookies are captured
- ▶ Is the targeted origin able to spot if we hijack the authenticated session passing it to an instrumented browser?

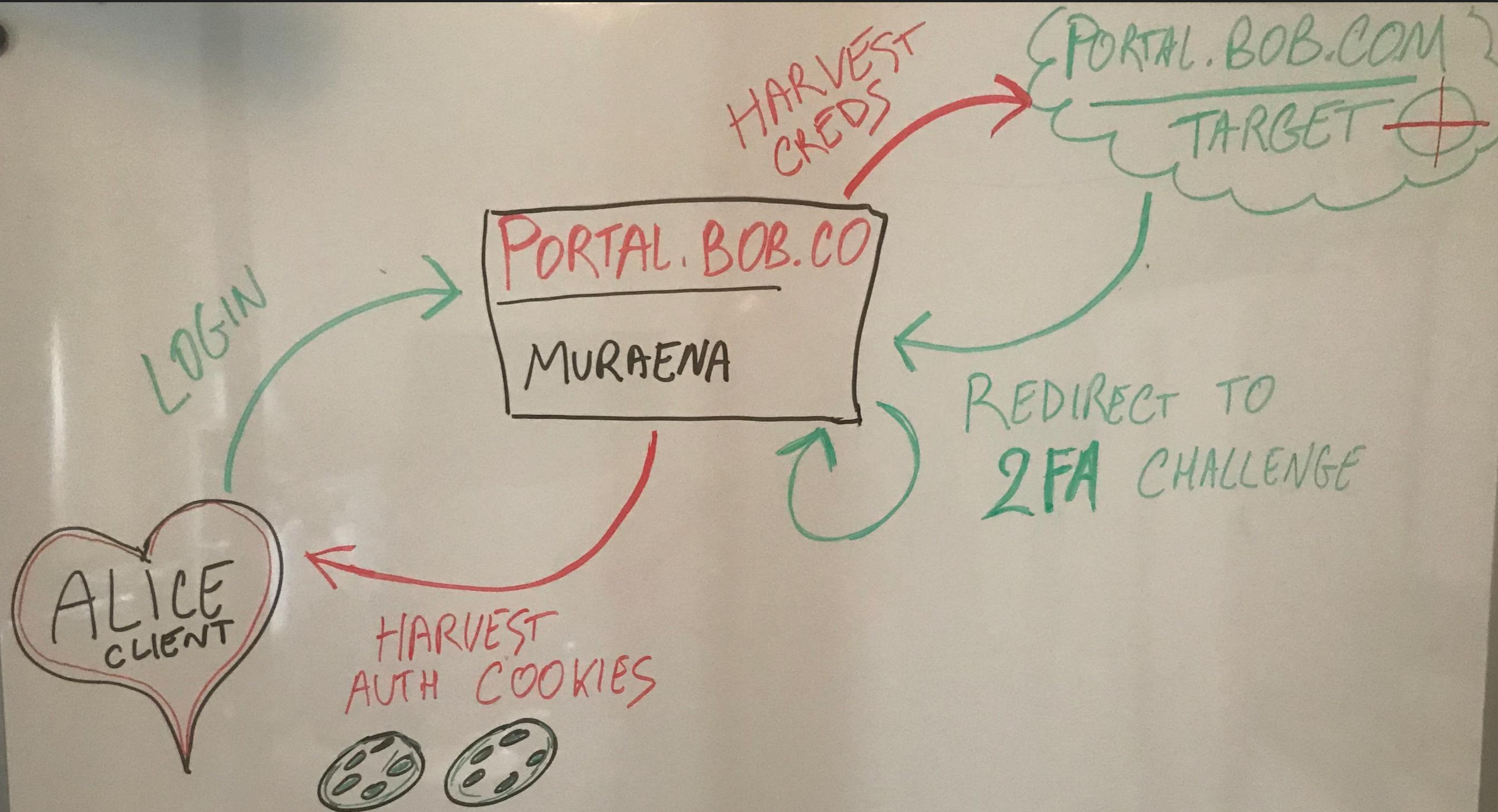
THE ART OF INSTRUMENTING STOLEN WEB SESSIONS

- ▶ Since all the traffic is passing through Muraena, credentials and session cookies are captured
- ▶ Is the targeted origin able to spot if we hijack the authenticated session passing it to an instrumented browser?
 - ▶ Usually **NO**
 - ▶ Additionally: the instrumented browser connection goes out via the same IP of Muraena, and the UA is changed to reflect the victim one.

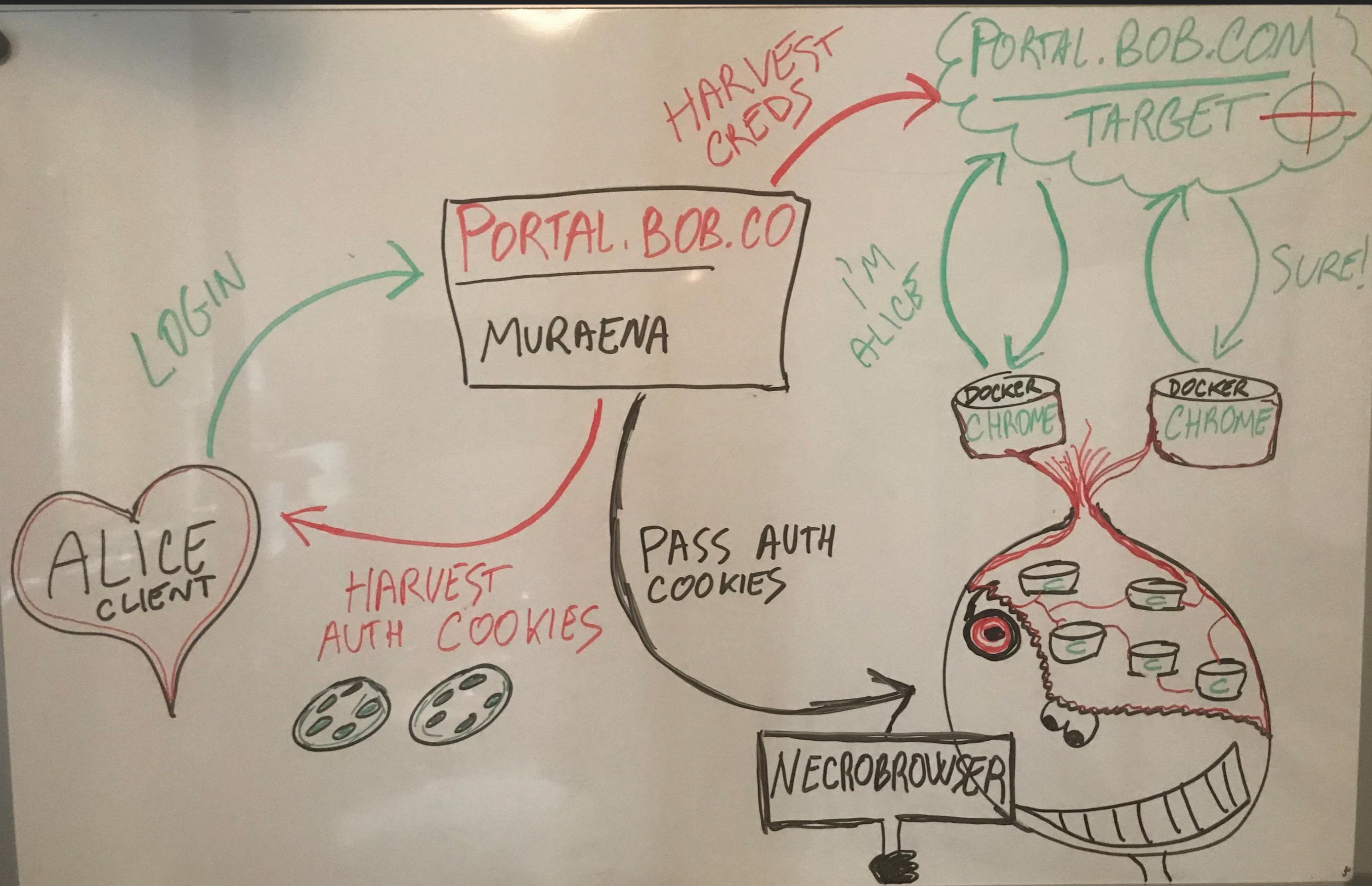
MURAENA WITH NECROBROWSER



MURAENA WITH NECROBROWSER



MURAENA WITH NECROBROWSER



NECROBROWSER

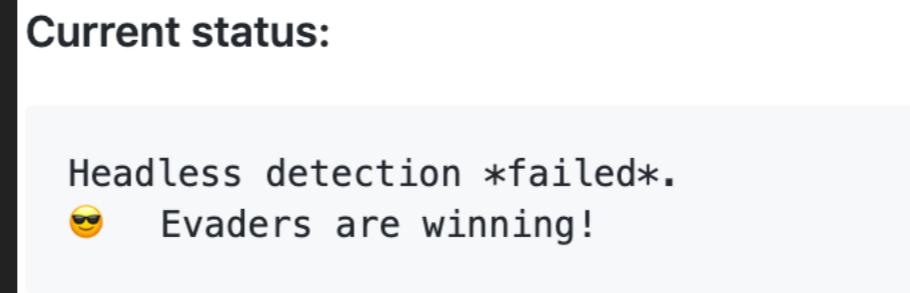
- ▶ NecroBrowser is a wrapper around chromedp
(<https://github.com/chromedp/chromedp>)
- ▶ Programmatically drive Chrome via Chrome DevTools Protocol (CDP)
- ▶ Exposed as a micro service that spawns dedicated Docker containers with Chrome
- ▶ Allows to keep alive as many session as your Docker server/cluster can support
- ▶ Can be scheduled to do repeated actions (dump emails every hour, read Slack messages every minute)

HEADLESS CHROME DETECTION



[http://antoinevastel.github.io/
bot%20detection/2018/01/17/
detect-chrome-headless-v2.html](http://antoinevastel.github.io/bot%20detection/2018/01/17/detect-chrome-headless-v2.html)

[https://intoli.com/blog/
not-possible-to-block-
chrome-headless/](https://intoli.com/blog/not-possible-to-block-chrome-headless/)



<https://github.com/paulirish/headless-cat-n-mouse>

HEADLESS CHROME DETECTION

- ▶ <https://intoli.com/blog/making-chrome-headless-undetectable/>
- ▶ Simply, it's not easy to detect a **non-human driven browser**

Test Name	Result
User Agent (Old)	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.39 Safari/537.36
WebDriver (New)	missing (passed)
Chrome (New)	present (passed)
Permissions (New)	pending
Plugins Length (Old)	5
Languages (Old)	en-US,en

AUTOMATING POST-PHISHING ACTIVITIES

- ▶ Examples could be endless, but the following are already implemented:
 - ▶ Disable GitHub Notifications, add SSH Key, download all repositories code
 - ▶ Disable GSuite Notifications, add Application Password, Dump Email
 - ▶ Upload arbitrary files to Dropbox, GDrive, Confluence
 - ▶ Screenshot and HTML dump all the things

FULL-CHAIN DEMOS

- ▶ Come se fosse Antani col video registrato anche per lei, senno' son moccoli anche se non partissero come ieri?



UPCOMING CHALLENGES

- ▶ Google:
 - Better protection against Man in the Middle phishing attacks
- April 18, 2019

However, one form of phishing, known as “[man in the middle](#)” (MITM), is hard to detect when an embedded browser framework (e.g., [Chromium Embedded Framework](#) - CEF) or another automation platform is being used for authentication. MITM intercepts the communications between a user and Google in real-time to gather the user’s credentials (including the second factor in some cases) and sign in. Because we can’t differentiate between a legitimate sign in and a MITM attack on these platforms, we will be blocking sign-ins from embedded browser frameworks starting in June. This is similar to the [restriction on webview](#) sign-ins announced in April 2016.

What developers need to know

The solution for developers currently using CEF for authentication is the same: [browser-based OAuth authentication](#). Aside from being secure, it also enables users to see the full URL of the page where they are entering their credentials, reinforcing good anti-phishing practices. If you are a developer with an app that requires access to Google Account data, switch to using browser-based OAuth authentication today.

FUTURE WORK

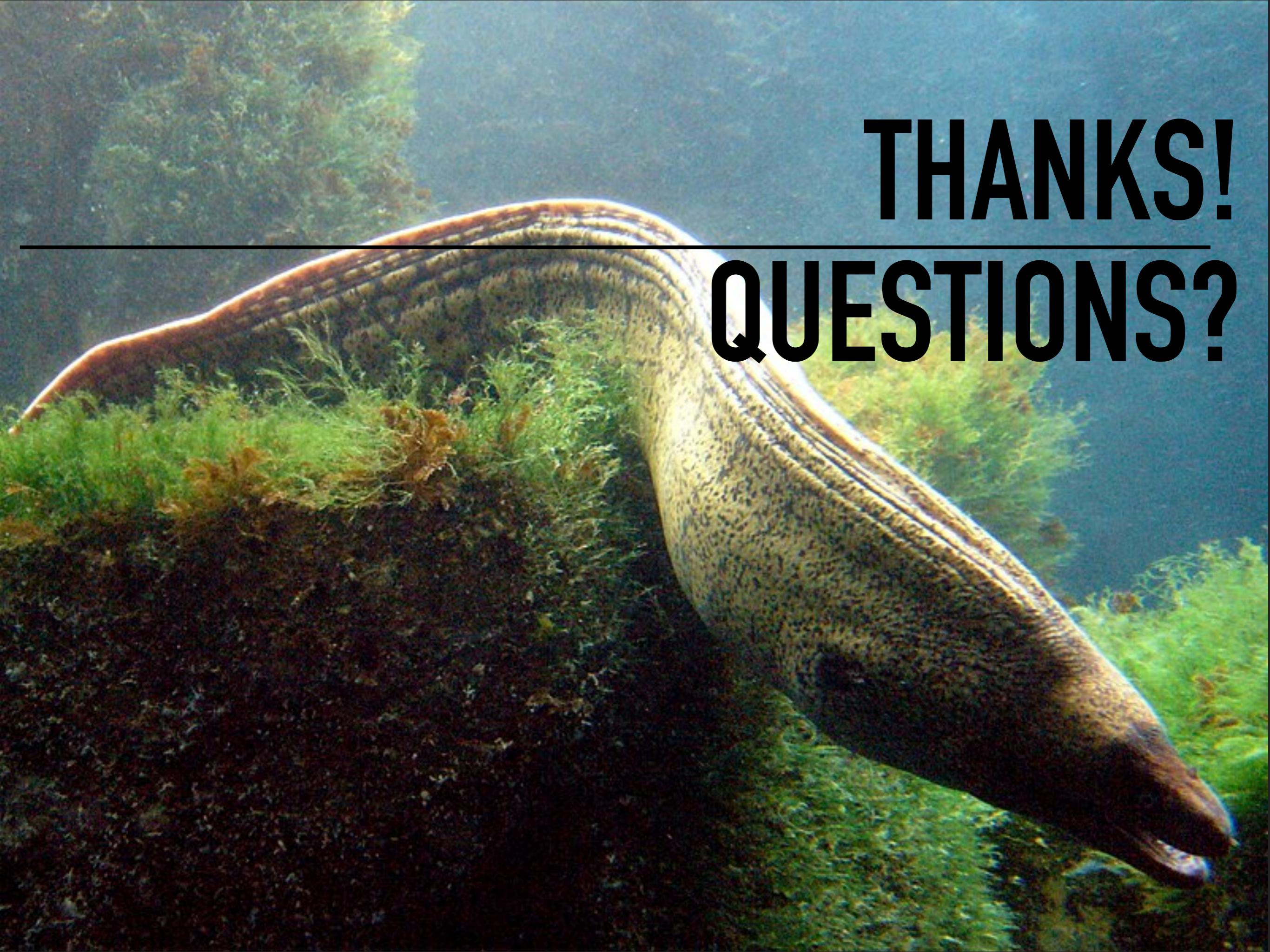


- ▶ Bettercap integration
- ▶ More browser automation fun:
 - ▶ Support for more commonly used web portals
 - ▶ Scaling tests, queuing instrumentation jobs
- ▶ Use browser instrumentation also for RECON/OSINT pre-phishing:
 - ▶ Scrape company X profile from LinkedIn/SocialNetworks using a real browser with a fake account

WHERE TO FIND THE CODE



<https://github.com/muraenateam>



**THANKS!
QUESTIONS?**
