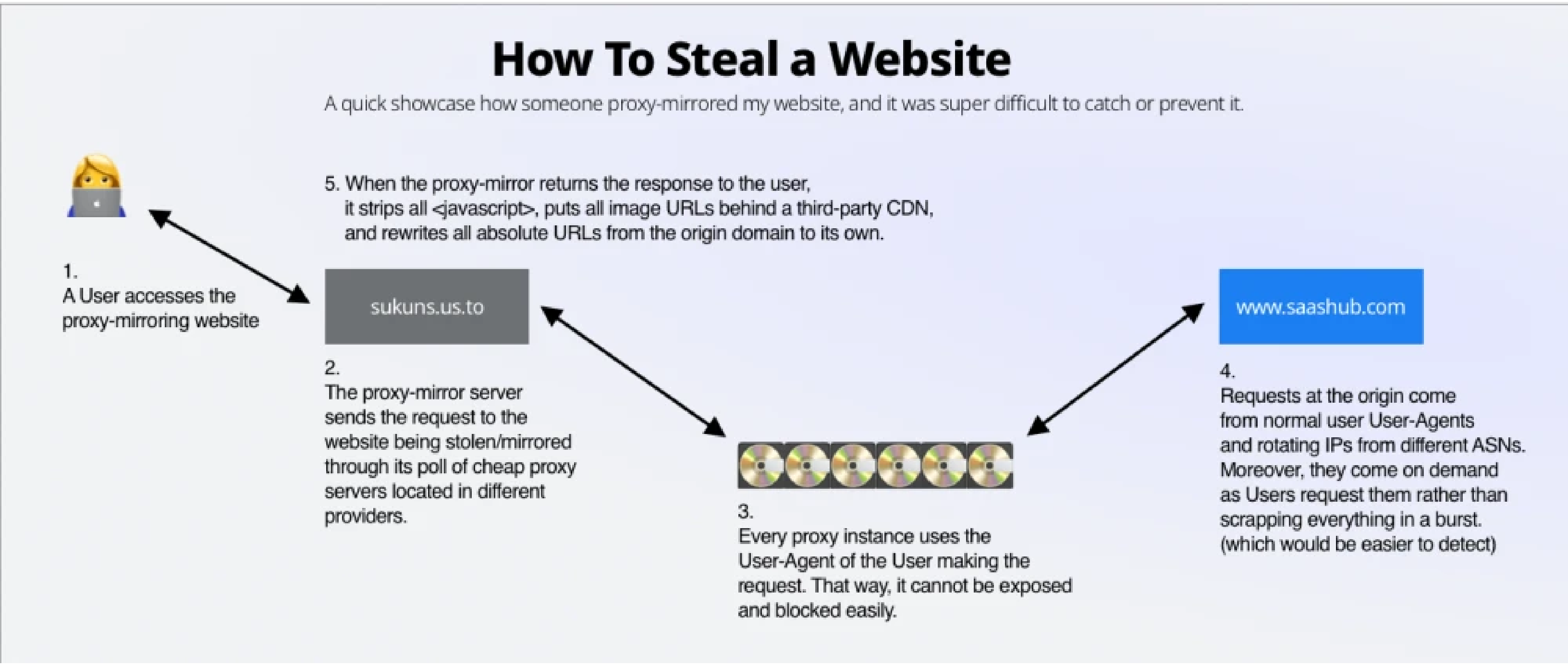

How To Steal a Website

A quick showcase how someone proxy-mirrored my website, and it was super difficult to catch or prevent it.

1.
A User accesses the proxy-mirroring website

2.
The proxy-mirror server sends the request to the website being stolen/mirrored through its poll of cheap proxy servers located in different providers.

3.
Every proxy instance uses the User-Agent of the User making the request. That way, it cannot be exposed and blocked easily.

4.
Requests at the origin come from normal user User-Agents and rotating IPs from different ASNs. Moreover, they come on demand as Users request them rather than scrapping everything in a burst. (which would be easier to detect)

5. When the proxy-mirror returns the response to the user, it strips all <javascript>, puts all image URLs behind a third-party CDN, and rewrites all absolute URLs from the origin domain to its own.

Stan Bright for SaaSHub
Posted on Dec 13

How to steal a website and how to prevent it

#webdev #tutorial #security #sysadmin

OK, this is going to be a quick showcase of how someone proxy-mirrored my website in a way that is very difficult to detect or prevent it. Moreover, I will share some of the helpful advice that was received from the Hacker News and /r/sysadmin communities.

Here it is the case

I noticed yesterday that one of my websites, SaaSHub, is not present on bing.com; however, its content is being served on a different and very suspicious domain name - sukuns.us.to. And, of course, I started digging my nginx logs - grepping for that domain name. All I could find was that domain name appearing as the referrer when accessing some images. Then, suspecting what might be going on, I started accessing some unpopular pages (on their domain name) and monitoring the logs for them. All requests to my server were coming from different IPs located in the US but with my browser's User-Agent. At that point, I was pretty sure what was happening - someone was proxy-mirroring SaaSHub and serving it on-demand on their domain name.

Here it is a simplified diagram of the case

I started thinking, brainstorming and searching for some generic advice on what I could do to prevent this...

Option 1 - block their IP. Unfortunately, that's not very effective in general, as they could easily change their IP address. What is more, in this particular case, they are using a very wide range of IPs coming from a variety of different ASNs (some of the cases I caught - AS55081, 35913, 36352, 21769, 52393, 394814, 55286). What is more, at first glance, collecting those IPs and ASNs isn't a trivial/quick task.

Option 2 - block them by user-agent - Again, not an option as they are copying the user-agent of the user making the request on their domain-name.

Option 3 - add a small piece of JavaScript that checks for the domain name, and if not www.saashub.com - redirect to it. This isn't a viable option in this case as the perpetrator is stripping all `<javascript>` tags and files.

Option 4 - use absolute URLs everywhere. Well, they are rewriting all mentions of www.saashub.com in links to their own domain name. So that doesn't seem like an option, too.

At that point, I felt helpless and asked for help on Hacker News & /r/sysadmin

Possible solutions and advice

i.e. what can you do if this happens to you... based on the suggestions and advice that I received on HN & /r/sysadmin.

1) File a DMCA to their host. I've sent a request to their free domain name provider so far.

2) Covert the whole website to JavaScript. Yup, that would work, but then no one without JS would be able to open SaaSHub, I'm guessing SEO will suffer too. So, this is not a path I'd like to take.

3) Block all images being accessed with a referrer that's not the expected domain name - I haven't implemented this one yet, but this could actually work in a generic way. I'm keeping this one in my toolbox of viable options.

4) Implement a honey-trap URL so that all their IPs could be exposed and blocked. This could work as well. "All" you have to do is add an endpoint that is not accessible by normal users - e.g. /honeytrap?some-random-param=rand-number (Note: you need the randomness as the culprit is loading each page once only and caching it after that). Then, add a cron script that will call that end-point on the proxy-mirror domain name and expose all their IPs. That way, we can block them in a relatively automated manner (as long as we know their domain name).

5) Add some hidden and random text to your pages. Why? So that you can search for that text later on and find out all the domain names that mirror your content.

Moral of the story

We are not absolutely helpless! It is unfortunate that we have to deal with it, but, as they say - it is what it is. At least we have a small toolbox of viable options on how to react and what we can try out.

Also, I guess there could be other things and means available to prevent and mitigate similar vicious actions. I'd be happy if you shared your experience and solutions in the comments here. You can find more advice within the discussions linked above, too. Nevertheless, whatever we (you and I) decide to do in similar cases, it's always a difficult and time-consuming battle. And I believe that bringing some awareness to this might help others resolve it quicker than I.

As of now, I think that the report to their free DNS provider has worked out, as their domain-name is not responding to an actual IP anymore. I will have to work on #5 from above, though, so that I can know when they move to a new domain-name.

p.s. if someone working at Microsoft or Bing is reading this, please could you help get SaaSHub indexed instead of the fake mirror sukuns.us.to. Thanks 🙇!

Top comments (0)

Code of Conduct • Report abuse

SaaSHub

Software Alternatives &amp; Reviews

More from SaaSHub

Replacing React with Preact. It was easy and worth it.
#react #preact #webdev #javascript

Postgres Trigram indexes VS Algolia
#webdev #programming #postgres #ruby