Using YouTube to steal your files
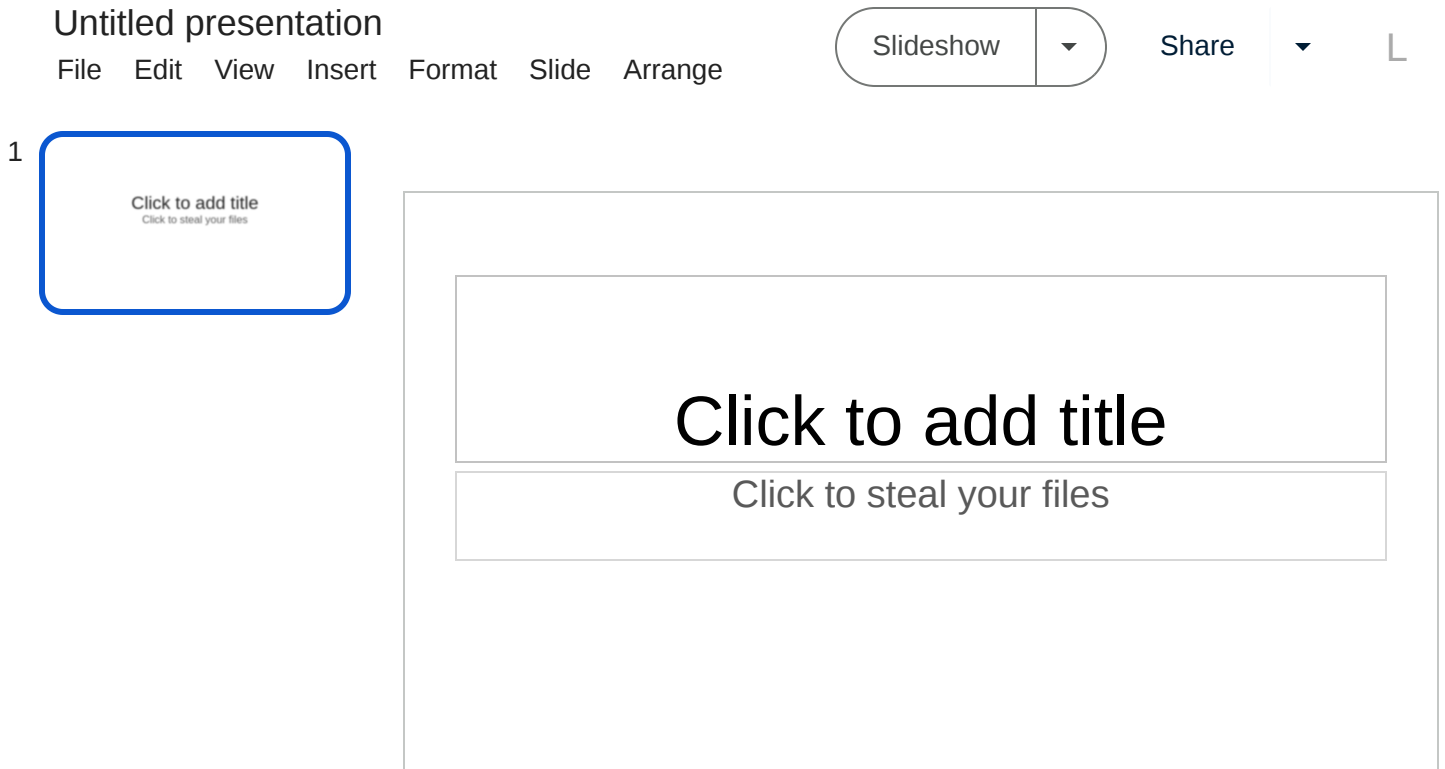
2024-09-19 ¦ infosec ¦ bug bounty

In my security research I often come across weird quirks and behaviours that aren't particularly useful beyond a neat party trick. It's always a good idea to keep track of them though, perhaps one day they'll be just the missing piece you need.

Untitled presentation

File   Edit   View   Insert   Format   Slide   Arrange

Slideshow ⌄   Share ⌄   L

1

Click to add title
Click to steal your files

Click to add title

Click to steal your files

Part 1: Cat videos

Who doesn't love cat videos?

Google Slides has this neat feature that lets you add YouTube videos to your presentations. Just open up the video picker, look for your favorite clip, and add it onto a slide.

What appears is an iframe that links to `www.youtube.com/embed/{VIDEOID}` with your cute cat video playing inside of it. Pretty neat! But can we do anything beyond just playing a video?

Looking at the network traffic, it seems like adding a video onto a slide will send Slides the videoid, which it then uses to construct the embed URL for the iframe. We can't control the full URL, just the videoid part. Can we still do something?

The obvious thing to try here is path traversal - if we change the videoid to `../`, the full url will be `www.youtube.com/embed/../`, which should turn into just `www.youtube.com/`, leading us straight to the YouTube home page. Let's try it!

To my surprise, it worked! We now have the YouTube homepage within this Slides iframe… or at least an error page representing it. YouTube, like most modern webapps, disallows framing most of its pages to prevent clickjacking attacks. Of course, the **/embed/** page is an exception because that page is intended to be embedded on other sites, but are there any other interesting **www.youtube.com** pages we could frame?
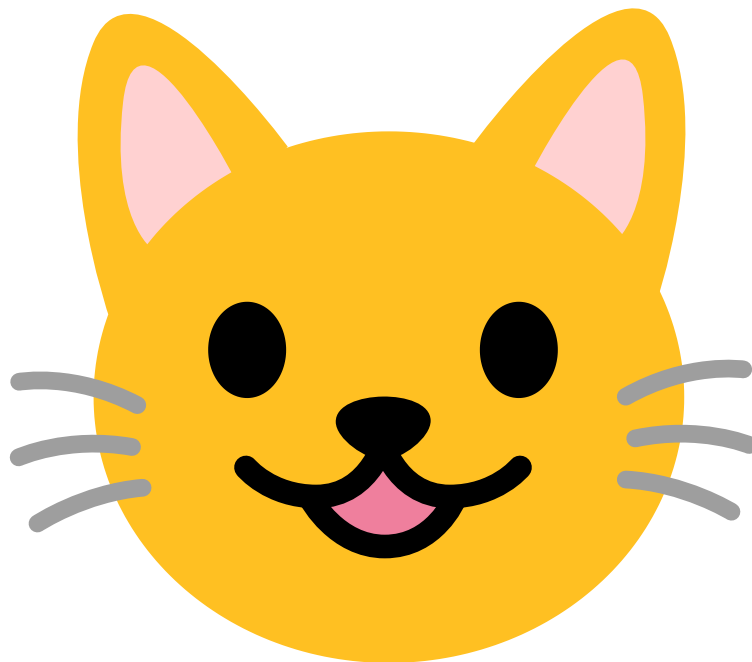
I looked into it for a bit, and found a bunch of framable resources on **/s/**. We can have stuff like YouTube's emoji and css/js source code inside of a presentation! Unfortunately, it doesn't seem very useful for now, it's just a fun trick we can do.

Part 2: Redirects

Open redirects are a genre of "vulnerabilities" that can redirect you to any other page. For example, visiting **google.com/url?q=https://lyra.horse**[1] will take you to **lyra.horse**. They are rarely considered to be real vulnerabilities because their impact is very limited - you'll just be redirected from one page to another.

Yet, as we're stuck in an iframe on **youtube.com**, an open redirect would be pretty lovely. Being able to navigate this Slides iframe to any website of our choice would let us do some very interesting stuff. So let's find one!

The first obvious place to look would be the external links around the site - such as the ones in video descriptions and comments. And indeed, clicking a link in the description of a video redirects us through a special **/redirect** endpoint:

```
https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbjdTaFRBeHRfSW95bkJDVmRGcl96VXV6
MkNmd3xBQ3Jtc0tuOVg2b2ZsQVV6V3hpaUJfdXB0UWY2Z1A1bE1sUjlQeHZ4WlVYSzNVUXZBcUF0RFYzNHhLazVUUUVFQM1Y5N3
VGZEV4bmtCVWhmYXRwY05KWlEyY0w3ZHBBdDY5SEtBa1hpQXBkalpqT3liYzFqYYVZxSQ&q=https%3A%2F%2Flyra.horse%2F
&v=tbYxAFHnzG0
```

The redirect works for now, but you'll notice it has a *redir_token* parameter - this parameter is some sort of a token for redirects that's unique to your session. If someone else opened the same link, they'd see this page instead:

⚏ youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbjdTaFRBeHRfSW95bkJDV...

Are you sure you want to leave YouTube?

The link is taking you to a site outside of YouTube (**lyra.horse**).

GO TO SITE   BACK TO YOUTUBE

It'd be difficult to convince someone to click through a page like that - and even so, we still wouldn't be able to use it inside of our cross-origin iframe due to it having the *x-frame-options* header set to *SAMEORIGIN*.

The next obvious place to look for open redirects is usually the authentication flow of a website - generally sites want to return you to the same page you were on before logging in. It's no different for YouTube, logging into a Google account takes you back to the page you were originally on. This is achieved through the **/signin** endpoint:

```
https://www.youtube.com/signin?action_handle_signin=true&app=desktop&hl=en&next=https%3A%2F%2Fwww.youtube.co
m%2F&feature=passive&hl=en
```

This endpoint does redirects without using a verification token! We can just specify an url of our choice in the *next* parameter and it'll work. Let's try it out with my website.

> **Sorry, we found some errors:**
>
> - **Invalid url forwarding parameter**
> - **Sorry, your login was incorrect.**

Oh, seems like it doesn't let us do an open redirect after all. Next I tried **google.com** - still the same error. I tried **youtube.com**… and once again, the same error?

I then realized that I had fogotten the subdomain - **www.youtube.com** does in-fact work with the redirect. And soon enough I discovered the redirects to work with any YouTube subdomain - **music.youtube.com** and **admin.youtube.com** both worked! We're still stuck on YouTube's domains, but at least we now have a bit more attack surface to work with.
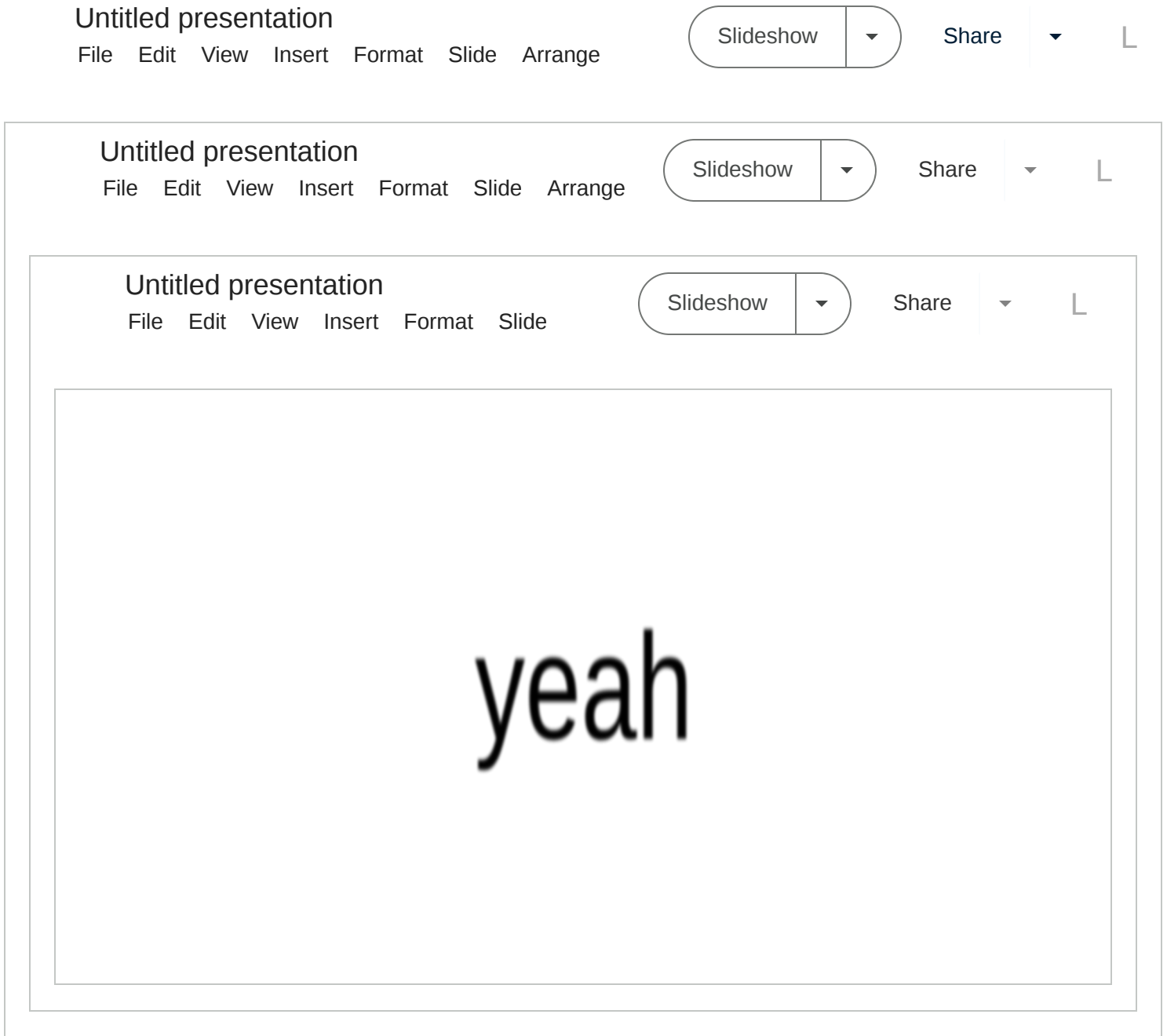
Part 3: Re-redirects

That **/signin** redirect wasn't the only one I found though - there was another one present on a different YouTube subdomain:

https://accounts.youtube.com/accounts/SetSID?ssdc=1&sidt=&continue=https%3A%2F%2Fwww.google.com&tcc=1&dbus=EE

This one seems to be for Google account logins. For example, if you log in on **google.ee**, you'd get redirected through **accounts.google.com** and **accounts.youtube.com** to update the cookies on both of those domains. I played around with it a little and found that while it once again wasn't a full open redirect, it did allow a variety of Google's own domains in the *continue* parameter, including services such as Docs.

If we could redirect our iframe to **docs.google.com** it'd open up a lot of possibilities. Google Docs is built in a way where most of its pages set the *x-frame-options* header to *SAMEORIGIN*, meaning that we're not supposed to be able to frame those pages on other websites. However, with such a redirect in place, we'd end up with a same-origin iframe within Slides, allowing us to frame pages we're not supposed to, and do cool stuff to them!

Let's try chaining our previous path-traversed **/signin** redirect to the new **accounts.youtube.com** one and see if we can make it embed Docs pages within itself.

Untitled presentation

File   Edit   View   Insert   Format   Slide   Arrange

Slideshow   ▾   Share   ▾   L

Untitled presentation

File   Edit   View   Insert   Format   Slide   Arrange

Slideshow   ▾   Share   ▾   L

Untitled presentation

File   Edit   View   Insert   Format   Slide

Slideshow   ▾   Share   ▾   L

yeah

And meow - Docs inside Docs!! So epic!

Part 4: Okay but what now?

So we have Docs inside of Docs, which is incredibly fun for a few minutes, but can we actually do anything useful with this? I played around with the Docs homepage for a bit, but the only interesting interaction I managed to find is deleting a document, and that's something you could restore from trash anyways. We'll need to find something more impactful on the Docs domain.

You might think that the document editing pages themselves would be useful, but those pages already have protections in place because they're already (intentionally) framable on external websites. If a page detects that it is within an iframe, it'll disable a lot of the dangerous functionality, such as the sharing options of the document.

This part here is what actually took me the longest to figure out. I spent a while looking for anything interesting on the **docs.google.com** domain to frame and clickjack. Looking through the Wayback Machine[2] and trying various Google

dorks[3], I kept finding a bunch of old endpoints that would've been useful in the past, but now just redirect to Google Drive, which we cannot frame.

Going through link after link, I eventually stumbled upon this url: docs.google.com/file/d/{ID}/edit . This page lets us preview and perform actions (such as sharing) on Google Drive files, and unlike the other links I found earlier, it stays on the **docs.google.com** domain instead of redirecting to Drive. And not only does it work with Drive files, it also works with folders and other such entities (such as Google Sites pages). You could even open up your Drive's "Root" folder[4] with it!

docs.google.com/file/d/0ALK4w9WgXcQUUk9PVA/edit

Root   Open with   Share   ▾   L

No preview available

The page has a share button that stays enabled even within an iframe. If we can trick someone into clicking the Share button, typing in our e-mail, and changing the permissions on some important folder, we'll gain access to it.

Part 5: But can we?

But let's do a reality check - can we *really* trick someone into performing all those actions? Maybe, if we try hard enough, but even with all our iframing and clickjacking abilities it's going to take a lot to convince someone to do all that. I don't think the VRP panel[5] would be very impressed with *this* much reliance on social engineering. We must find a way to make it more convincing - ideally condensing it down to just a single click.

Thinking of ways to improve the attack, I remembered the feature in Drive that lets you request access to other people's documents. Doing so sends out an e-mail with a cool little button to immediately manage the permissions.

Share request for "Secret Folder"   Inbox ×

Lyra Rebane (via Google Drive) <drive-shares-dm-noreply@google.com>   Sep 19th, 2024, 10:30 AM   ⋮
to me ▾

Share a folder?

L
Lyra Rebane (lyra.horse@gmail.com) is **requesting access** to the following folder:

hi pls give access kthxbye

📁 Secret Folder

Manage sharing

The button in that e-mail links to https://drive.google.com/drive/folders/{ID}?usp=sharing_esp&userstoinvite=lyra.horse@gmail.com&sharingaction=manageaccess&role=writer&ts=66e724ba , which when opened, pops up the Share dialog with a notification of the request. Of course, that's a Drive link, not a Docs one, but I tried copying all of the query parameters over to our Docs link and that seemed to do the trick!

docs.google.com/file/d/1sHy3aQXsIlnOCj-mBFxQ0ZXm4TzjjfFL/edit?usp=sharing_esp&userstoinvit...

Share "Secret Folder"   ⊙

Lyra Rebane asked to be an editor   Review   ✖

┌──┐
│ Add people, groups, and calendar events   │
└──┘

People with access

L   Lyra Rebane (you)   Owner
     lyra.horse@gmail.com

General access

🔒   Restricted ▾
     Only people with access can open with the link
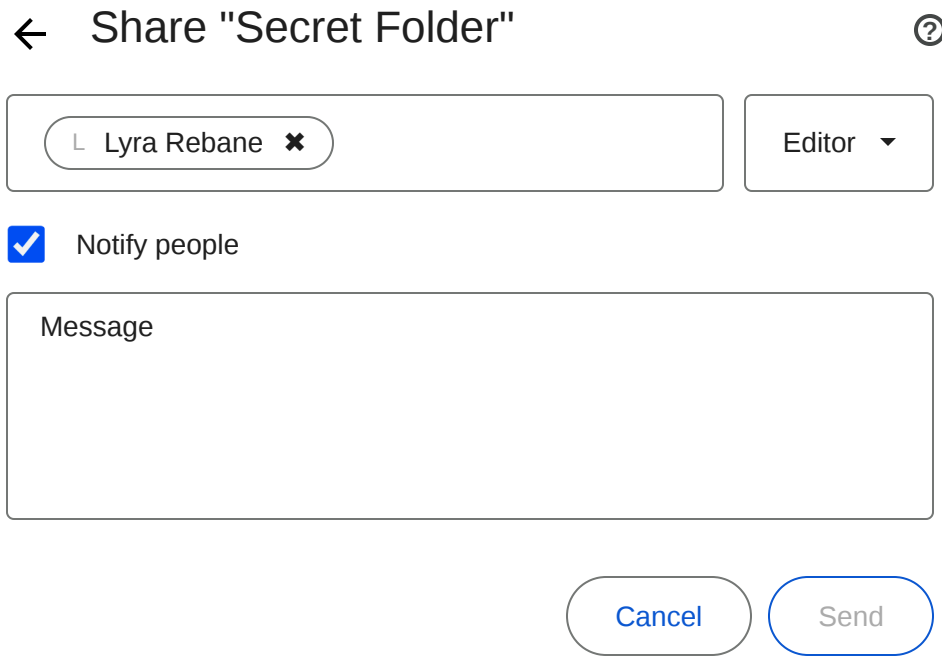
(Copy link)   (Done)

In its current state, this page requires us to make two clicks to complete the attack - first a click on the "Review" label, and then a click on the "Share" button (try clicking "**Review**" above). That's already quite good, but I still *really* wanted to get the entire attack down to just one click.

I pulled out my DevTools and began digging through the JavaScript of the page to see how the query parameters are handled. As a simple test, I started off with just the *userstoinvite* query parameter.

⚏ docs.google.com/file/d/1sHy3aQXsllnOCj-mBFxQ0ZXm4TzjjfFL/edit?userstoinvite=lyra.horse@gm...

← Share "Secret Folder" ⑦

⌊ Lyra Rebane ✖ | Editor ▾

✔ Notify people

Message

Cancel   Send

And wow!? I had accidentally stumbled upon the perfect share dialog URL. For some reason, leaving out all the other query parameters makes the share dialog just auto-fill the e-mail field from the query parameter, defaulting to giving out *Editor* permissions.

Pretty much all we need to do here is convince someone to do a single click on the ambiguously labeled "Send" button, and we're set!

Part 6: Re-re-redirects

I began putting the attack together, combining all the cool tricks we've come up with so far.

1. We first take cool little docs invite url.
   https://docs.google.com/file/d/1sHy3aQXsllnOCj-mBFxQ0ZXm4TzjjfFL/edit?userstoinvite=lyra.horse@gmail.com
2. Then we put it inside the **accounts.youtube.com** redirect.
   https://accounts.youtube.com/accounts/SetSID?continue=https%3A%2F%2Fdocs.google.com%2Ffile%2Fd%2F1sHy3aQXsllnOCj-mBFxQ0ZXm4TzjjfFL%2Fedit%3Fuserstoinvite%3Dlyra.horse%40gmail.com
3. Then we put *that* into the **youtube.com/signin** redirect.
   https://www.youtube.com/signin?next=https%3A%2F%2Faccounts.youtube.com%2Faccounts%2FSetSID%3Fcontinue%

4. And finally, we turn it into a path traversed "videoid" we can embed in our slides.

../signin?next=https%3A%2F%2Faccounts.youtube.com%2Faccounts%2FSetSID%3Fcontinue%3Dhttps%3A%2F%2Fdoc
s.google.com%252Fa%252Fa%252Ffile%252Fd%252F1sHy3aQXsIlnOCj-mBFxQ0ZXm4TzjjfFL%252Fedit%253Fuserstoinv
ite%253Dlyra.horse%2540gmail.com

And there we go! I threw it in my slides and…
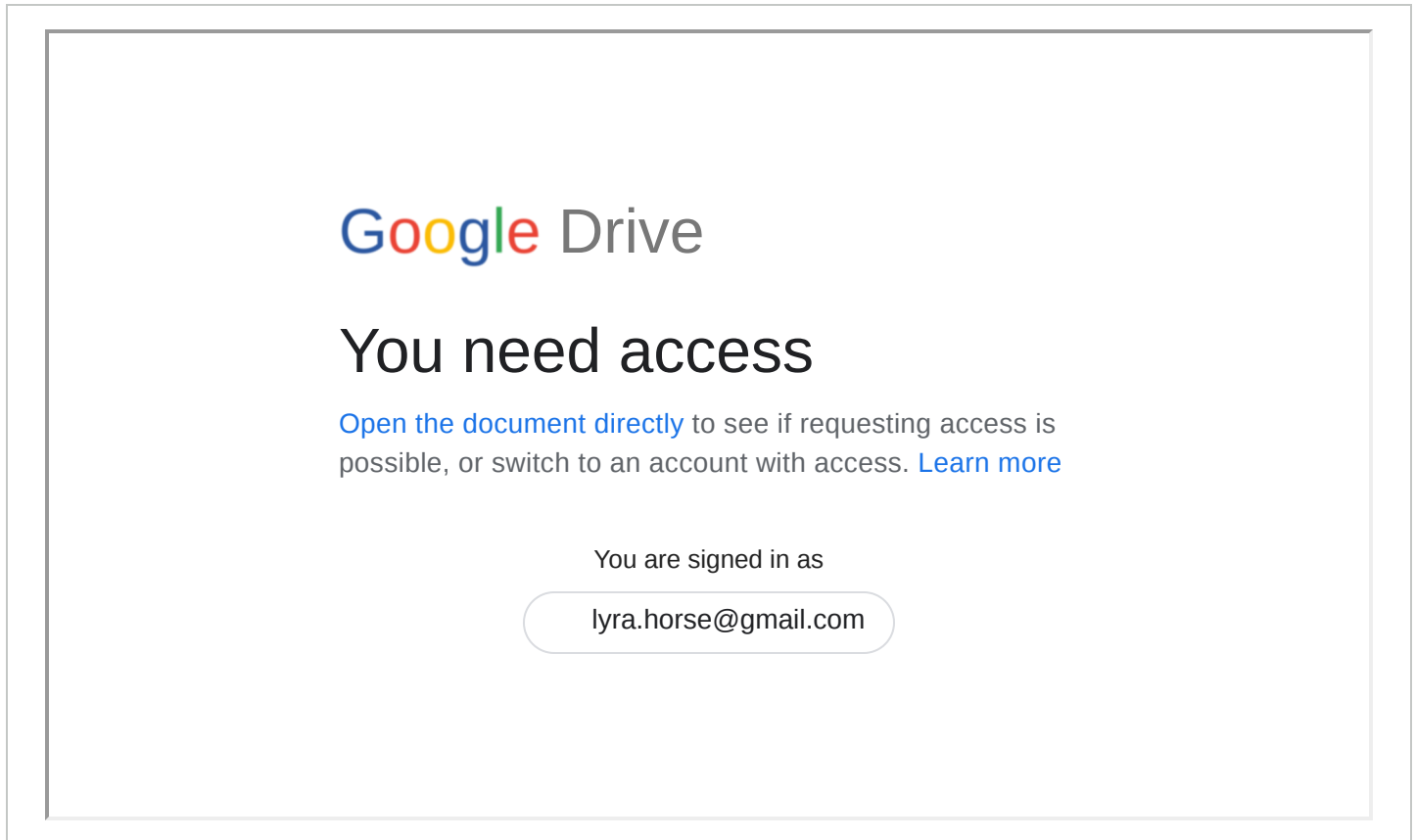
…it didn't work, why?

It seems like Docs has some sort of a mitigation in place that prevents me from using a cross-site redirect for the file page within an iframe. More precisely, it checks for the *Sec-Fetch-Dest* and *Sec-Fetch-Site* headers, and if they're both set to *iframe* and *cross-site* respectively, we get a 403 back. Pretty weird.

I got the opportunity to chat with a couple security people from Google, so I asked about this behavior, and it seems like this is some sort of a mitigation to prevent cross-origin framing on the server-side. I'm still not entirely sure as to what threat scenario it'd be useful in, but the idea is that an iframe can tell whether it's on a same-origin page or not from just the *Sec-Fetch-Site* header. On a cross-origin page, the header will *always* be set to *cross-site,* even if the redirect within the iframe is same-origin.

Of course, that could be detected more reliably on the client-side with JavaScript and whatnot, but the headers are the only way for a server to tell *before* sending out a response. A side-effect of the server-side detection is that even though both our frames are same-origin, a cross-origin redirect within the iframe will still end up with the *cross-site* header. To bypass *that*, we need to perform a same-origin redirect inside of the iframe.

To put it simply, we're currently doing:

accounts.youtube.com **(cross-site)** → docs.google.com/file/d/.../edit **(403)**

so to bypass that, we want to chain a redirect like this:

and it should work! But we have to find something that'd work for that part in the middle. And lucky for us, I had already spotted something like that in my googling earlier.

It seems like there's an old legacy GSuite URL format of **docs.google.com/a/<domain>/…**, which probably did something useful years ago (edit: and still does[6]), but these days just disappears when you open an URL. If you're logged out, you must find some working donor URL to use, such as **/a/wyo.gov/**[7], but logged in you can even do **/a/a/** and it'll just work.

Here are a couple of example URLs to try out.

This one should work regardless of your login state:

 https://docs.google.com/a/wyo.gov/file/d/10LlimFowOJ_noDrJsv4CnRgU8XoUKRAa6YjTeJFrs70/edit

And this one requires that you be logged into any Google account:

 https://docs.google.com/a/a/file/d/10LlimFowOJ_noDrJsv4CnRgU8XoUKRAa6YjTeJFrs70/edit

Both will end up redirecting to https://docs.google.com/file/d/10LlimFowOJ_noDrJsv4CnRgU8XoUKRAa6YjTeJFrs70/edit .
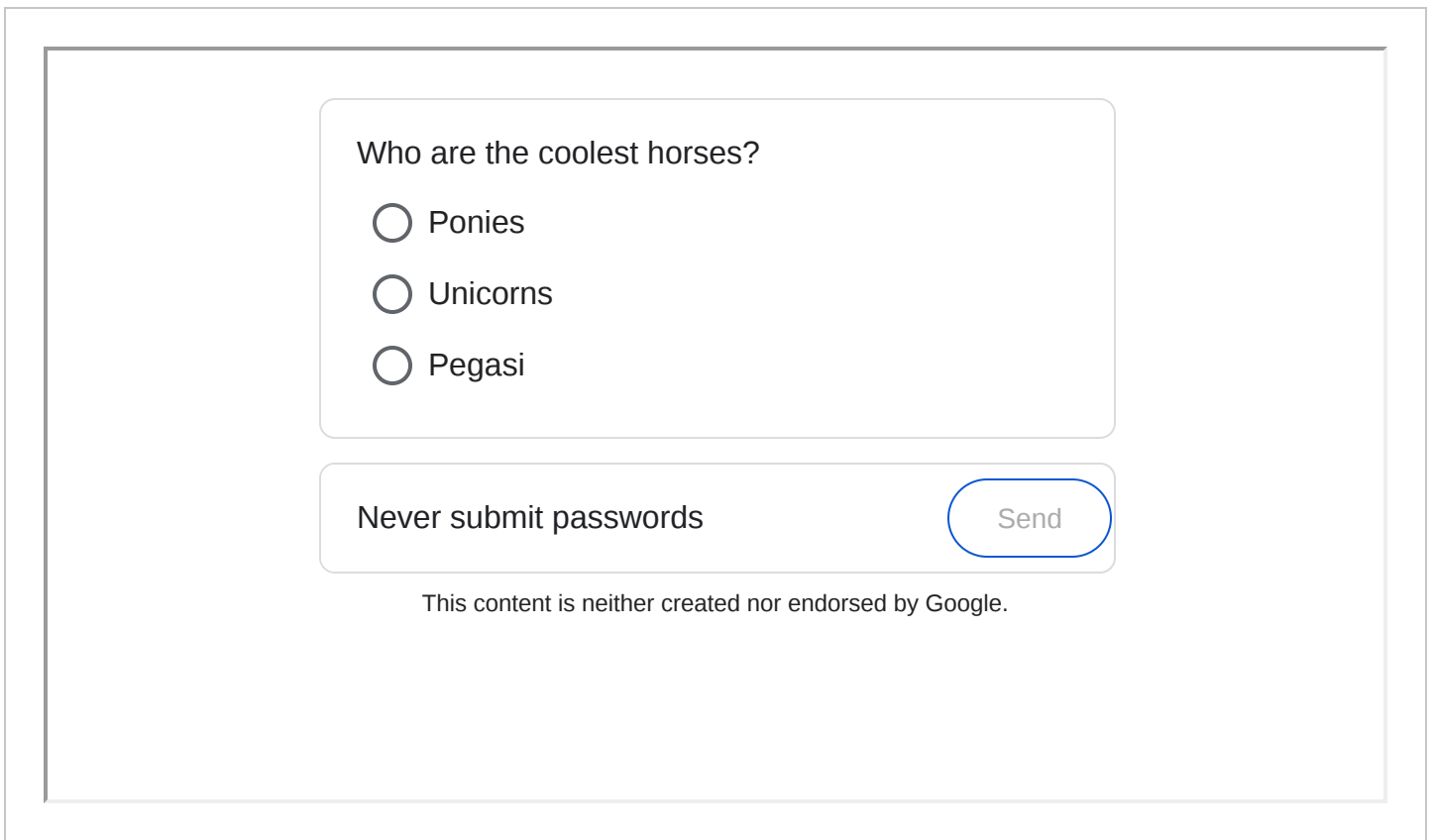
With that figured out, let's throw the **/a/a/** thing into our "videoid" from earlier: ../signin?next=https%3A%2F%2Faccounts.youtube.com%2Faccounts%2FSetSID%3Fcontinue%3Dhttps%3A%2F%2Fdocs.google.com%252Ffile%252Fd%252F1sHy3aQXsIlnOCj-mBFxQ0ZXm4TzjjfFL%252Fedit%253Fuserstoinvite%253Dlyra.horse%2540gmail.com
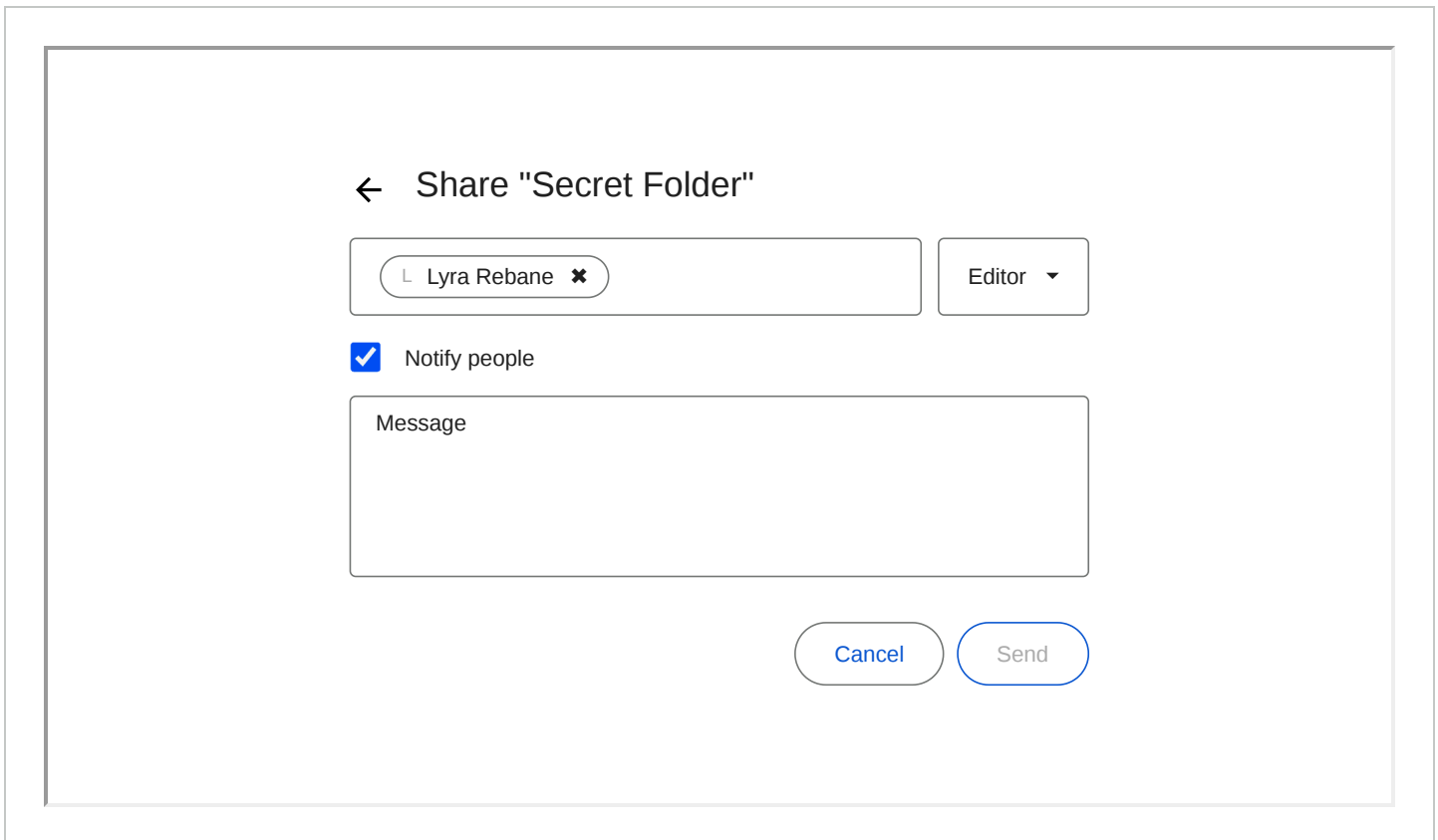
And it works!

Part 7: Finishing touches

With our share dialog inside a presentation, all we need to do now is cover it up with other stuff to make it look presentable. Since all we need to do here is get someone to click the "Send" button, I decided to make my demo look like Google Forms.

Who are the coolest horses?

⚪ Ponies

⚪ Unicorns

⚪ Pegasi

Never submit passwords   Send

This content is neither created nor endorsed by Google.

And we're done! It looks like a Google Forms page, but it has a "cutout" for the "Send" button in the Share dialog below. If clicked, it'll immediately share *Editor* permissions for the targeted file/folder with whatever e-mail we specified. To send this attack to someone we can replace the **/edit** with **/present** in the Slides url to have it open and "play" the slide direcly.

And there we go, a one-click clickjacking attack that chains a Google Slides YouTube embed path traversal to three separate redirects to gain editor access on a Drive file/folder!

I reported this vulnerability chain to Google on the 1st of July 2024, and got it triaged & confirmed on the same day! 10 days later, on the 11th of July, the VRP panel awarded me with a reward of **$3133.70** + **$1000** bonus, totalling **$4133.70**. Sweet!

afterword

thank you for reading, you're awesome!!

i tried to keep this writeup condensed because i'm also presenting my research with additional story elements at bsides tallinn 2024 the same day this blogpost goes out. i hope it goes well! i'm not sure when the bsides talk recordings will be released (keep an eye on this channel), but for now you can check out the slides!

as with my previous posts, everything on the page is just html/css crafted with love. no images, javascript, or other external resources, and just 31kB gzipped (that's 5 seconds over dial-up)! it takes a lot of time and effort compared to just throwing screenshots on the page, but i think it's really fun to have a blogpost come to life like that, with interactivity and all. and it's responsive!

i hope this writeup is conherent and interesting to read, the attack chain involves quite a few elements so the article is all over the place at times, you can always feel free to ask me any questions if anything's unclear ^^

love you all <3!

Discuss this post on: twitter, mastodon, lobsters (rip cohost :c)

1. This specific example will probably display a warning - but let's just pretend it doesn't. ↵

2. The Internet Archive allows listing all archived URLs for a domain, quite handy for recon. ↵

3. Dorks are the various search operations and tricks you can use on Google, such as `site:docs.google.com` or `inurl:document.` ↵

4. Every Google Drive file and folder has an ID associated with it, and your entire drive's Root folder is no exception! Want to find yours? Open Drive's page with DevTools open, and then search for `9PVA` in the network requests. ↵

5. The VRP is Google's bug bounty program, and its panel is a group of people who decide how much $$$ you'll get for a bug. ↵

6. advaith let me know that /a/domain urls automatically switch your account to the one under the domain, so they're still handy if you use multiple accounts. ↵

7. I'm using this domain as an example because it's short and came up a lot in my Google searches, but there isn't anything special about it, you can use other gsuite domains too. In case anyone from the Wyoming goverment happens across this post - no, this isn't touching your IT systems in any way, it's only affecting Google's systems and they're already aware of and working on the topics discussed in this blog post. ↵