Saul Johnson

I Think You Left Your Redirect Open

A (Very) Brief Introduction

I’m Saul, a software verification researcher here at Teesside University.

I’m mainly working with formal methods around password strength/cracking but security research is fun too!

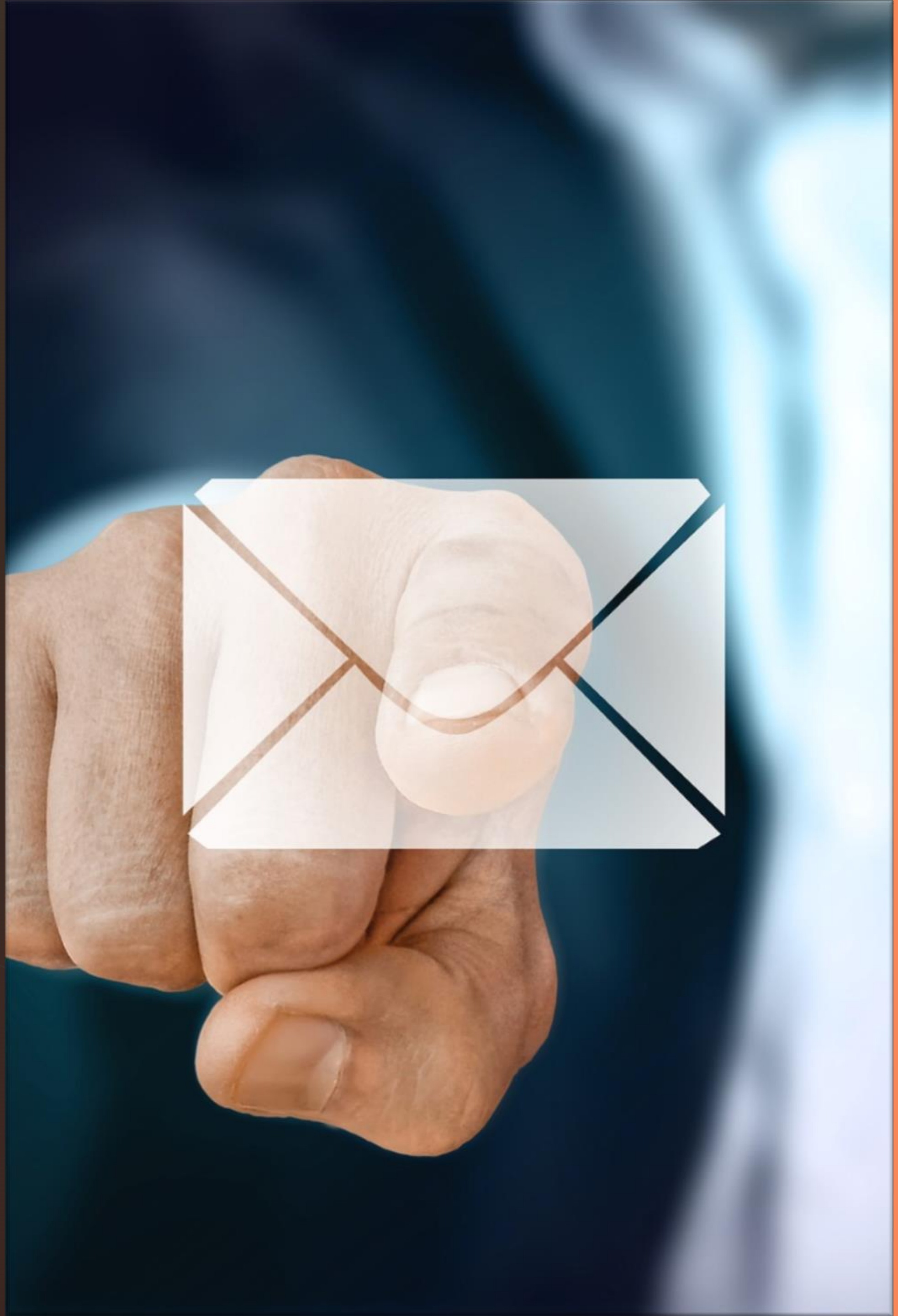
GitHub: @lambdacasserole
Twitter: @lambdacasserole
Web: https://sauljohnson.com

What is it?

Phishing

Before we begin…

I’m just going to check my e-mail real quick. Bear with me…



So… what just happened?

- Our credentials have just been stolen and our Google account has been compromised.
- Did anyone spot the point at which this happened?
- A few points to note:
  - No clever DNS/SSL trickery was involved, we *did* sign in through the legitimate Google login page.
  - We really ended up on the real google.com.
  - This attack can absolutely be carried out over the open internet today (it hasn't been fixed).
- Google is absolutely aware that this can be done. This class of attack is well-known.

Now, we'll play the attacker…

It's time to take a look at this attack again from the perspective of the party carrying it out…

Unvalidated Redirect/Forward

- Somehow we were able to insert our own step into Google’s login process. How did we do this?
- We used a well-known vulnerability called an *unvalidated redirect/forward*.
  - The Google login page takes a URL parameter `continue`. On successful login the user will be redirected to the URL passed as this parameter.
  - For example, this will redirect us to the Google home page after logging in successfully: `http://accounts.google.com/?continue=google.com`

But This Redirect Is Validated!

This redirect is validated, however. As far as we can tell, URLs passed to the `continue` parameter need to match the following regular expression for the redirect to go ahead:

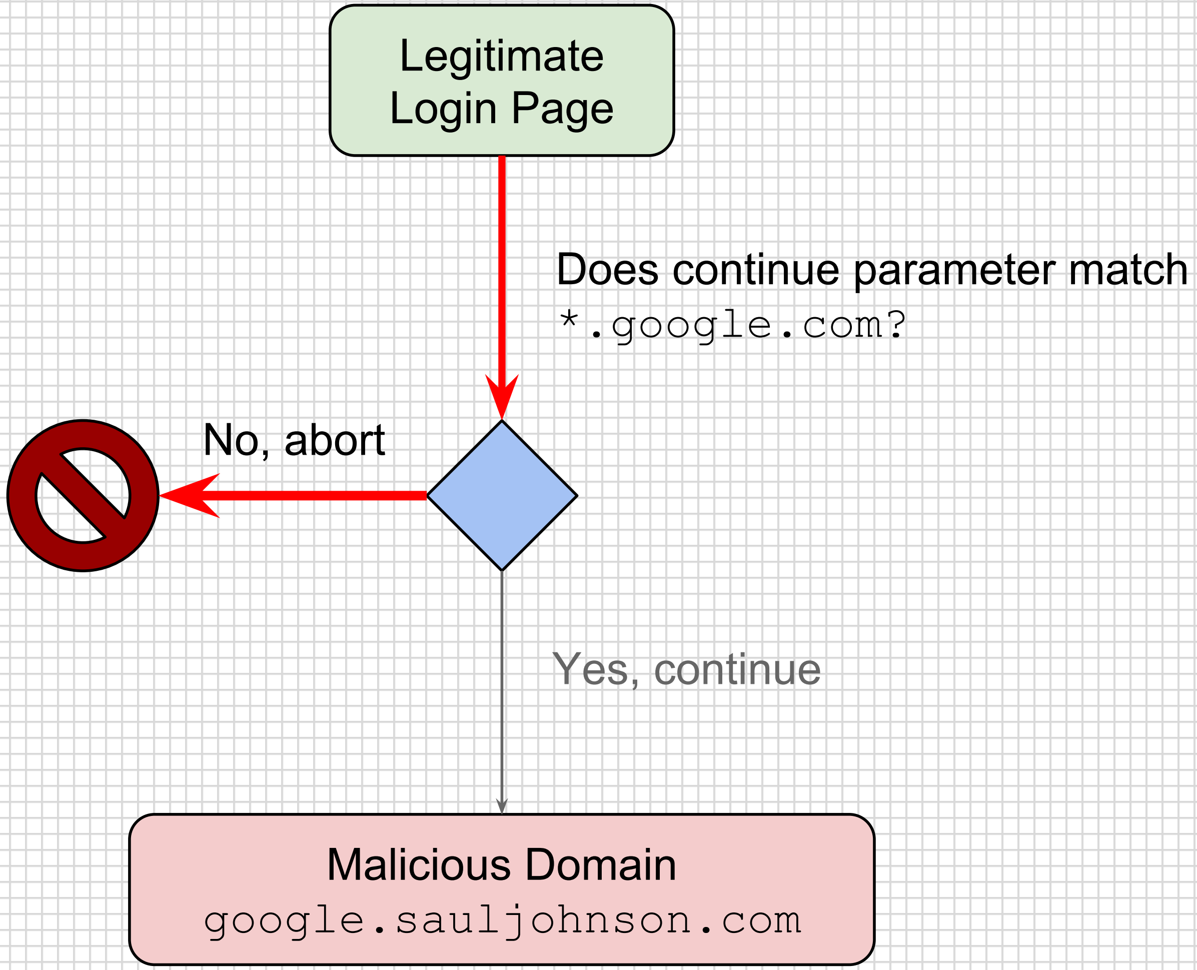
```
*.google.com
```

What’s The Problem Then?

- The problem is that a regular expression containing wildcards (*) is used as the whitelist.
- With such a vast array of web services provided by Google at the `google.com` domain, Google must make sure that all redirects/forwards from that domain to any client-specified URL *also use that whitelist*.
- This is still not enough (user-hosted content is also present at `docs.google.com` and `drive.google.com`) but it is a start.
- If just one unvalidated redirect at any `google.com` domain is present, this whitelist becomes useless…

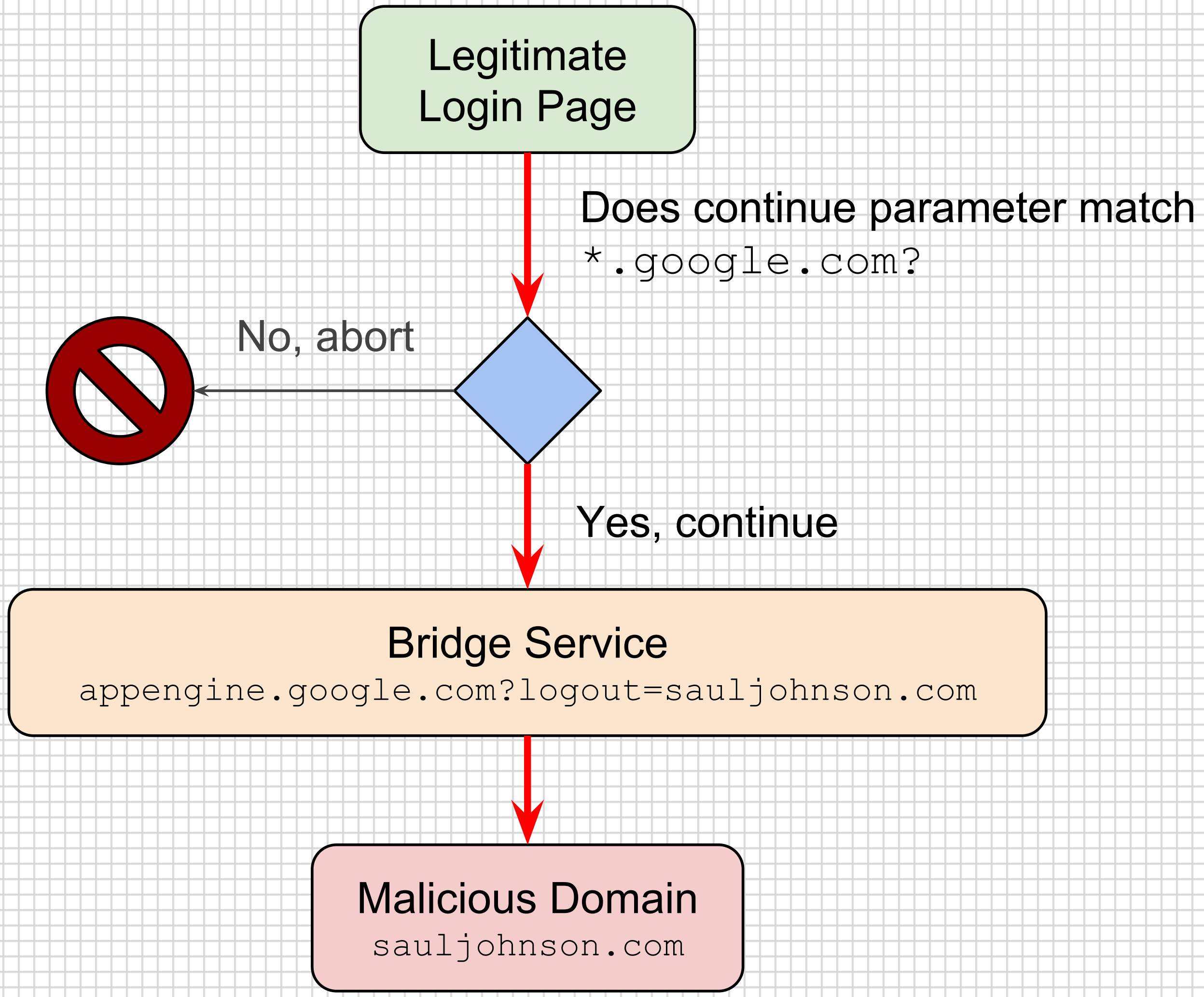
What’s The Problem Then? (contd.)

On the right is a flow diagram showing how Google probably intended this to work. Indeed, it is how it works if we try to redirect directly to a malicious domain. We can get around this, however…



What’s The Problem Then? (contd.)

Here lies the issue. We can use a Google open redirect that *doesn’t* use a whitelist as a “bridge” service to (almost) silently redirect the user straight from the legitimate login page to a malicious website.



Protecting Users Against This Threat

- It is possible to avoid falling victim to this attack by remaining aware of the domain you’re currently on at all times.
- Two-factor authentication would also prevent the attacker from accessing our Google account if our password was compromised.

Protecting Users Against This Threat (contd.)

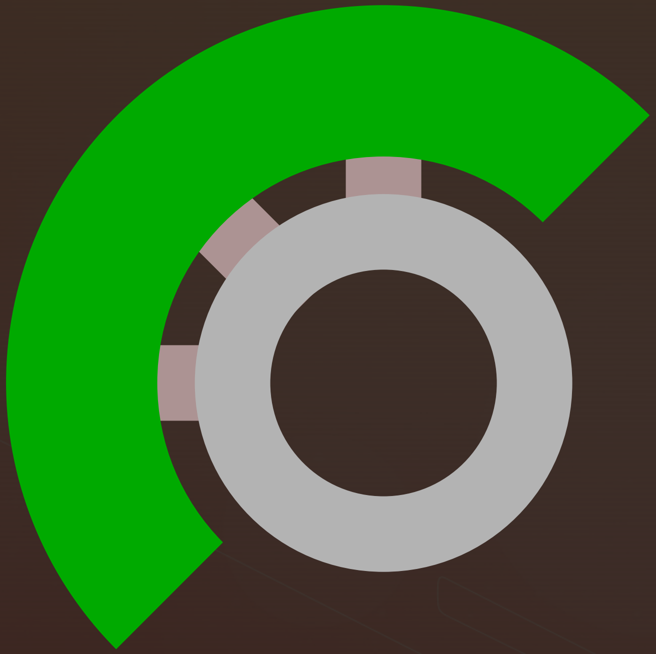
- This ‘manual’ protection has certain disadvantages:
  - Users with no understanding of DNS can’t currently be assumed to know that `google.sauljohnson.com` is not a Google domain, but just a subdomain above `sauljohnson.com`.
  - Not all browsers (particularly on mobile) show the URL at all times (I’m looking at you iOS Safari).
  - Two-factor authentication would keep our Google account safe, but our password would still be compromised.

Automating Protection

- Is it possible to protect against this class of attack automatically? That's another talk and probably a short research paper.
- The short answer is yes, but:
  - We require a *trigger event* like a login form submission to alert us that this attack may be beginning.
  - We need to be able to tell if we're redirected away from the domain suspiciously quickly after a trigger event.
  - We think that it probably gets much more complicated than this. More research is required.

That Being Said… Prototype!

- We have developed a prototype Chrome extension that protects us against this attack.
  - It's called *Ordinator* for **o**pen **redi**rect termi**nator**
  - It works in just the way we described earlier. A login event followed suspiciously quickly by a change in domain triggers a warning.
  - It's not currently available.
- Demonstration time…



An Important Acknowledgement

- Though I've played around with it quite a lot, I wasn't the one that originally discovered this vulnerability.
- That was a security researcher called Aidan Woods, who helped enormously with this talk and all those before it on this topic.

Website: https://www.aidanwoods.com/

Twitter: @aidantwoods

Write-up (includes Google's response to this issue): https://www.aidanwoods.com/blog/faulty-login-pages

Thank you for your attention!

I’ll be around afterwards if you’d like to talk offline!