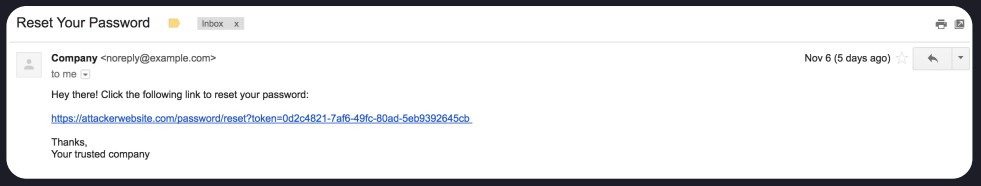
Don't Trust the Host Header for Sending Password Reset Emails

Jack Cable - December 13, 2017

Let's say you're building password reset functionality into your website. Since your website url might change in the future, you want to dynamically generate the link in the password reset email to match your server's url. No problem, a quick Google search returns this StackOverflow question for getting the server's url in PHP. You add `$_SERVER['HTTP_HOST']` to your code and think nothing of it.

If you haven't yet caught it, this is a really bad practice. The `Host` header is provided by the client, meaning an attacker could embed their own website in a password reset email, compromising any user's account. An attack scenario would look something like this:

1. The attacker identifies a target user's email address.
2. The attacker modifies the host header of the request to reset the target's password to their own domain.
3. The target receives the following email:



4. Trusting the company, they click the reset link. As the link is formed with the host header, this instead links to the attacker's website. When the target visits this site, their password reset token is sent to the attacker.
5. The attacker now resets the target's password using their password reset token.

Unfortunately, this attack isn't purely theoretical, and I have found it in 4 bug bounty programs over the past few months.

On a whim, I decided to test for this vulnerability on Mavenlink, following their public HackerOne program.

Mavenlink allows companies to register a subdomain of mavenlink.com to host their content, so it makes sense that they would need a way to dynamically determine the company's subdomain.

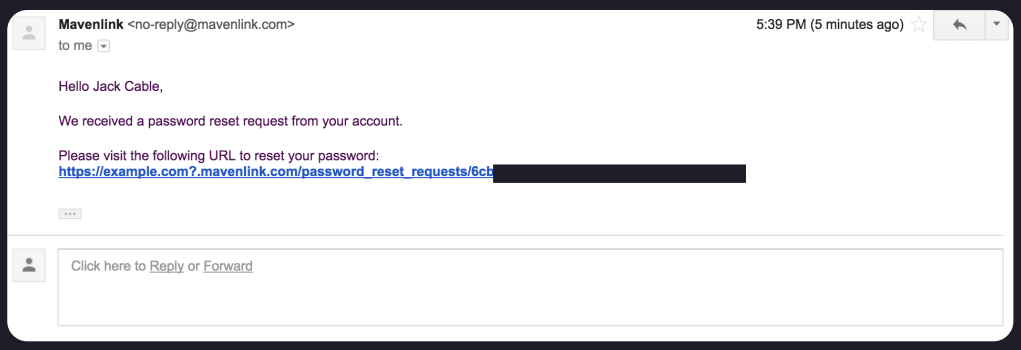
I quickly noticed that I could set the host header to any subdomain of mavenlink.com, which would be reflected in the email. However, changing the domain from mavenlink.com would return an error and not send an email.

At first, this seemed relatively secure. I couldn't find a way to inject my own domain — sure, I could add a random subdomain of mavenlink.com, but all of these redirected to the proper page.

I then started testing special characters. After a few tries, I hit gold: the server would accept a question mark in the header, so I could follow my domain with a question mark to make that the base url:

```
Host: example.com?.mavenlink.com
```

This allowed me to steal any user's password reset token from the email:



View the disclosed report at https://hackerone.com/reports/281575.

Conclusion

Don't trust the `Host` header for sending password reset emails. Ever. And you probably shouldn't be trusting it for anything else, either.

Instead, if you need to account for a dynamic url, it's a good idea to store the host as a server-side variable.