Introduction to SMTP Header Injection

CYBR

This ebook is a collection of lessons from our course, Injection Attacks: The Complete 2020 Guide. This version only contains the SMTP Header Injections section, and not other injection attacks for brevity. There is a separate ebook version that includes other injection attacks if you'd rather. We know some people may prefer the format of an ebook over a series of lessons online, or of written versus video lessons. In any case, we hope you enjoy it!

Table of Contents

Overview of SMTP Header Injections

Slides used in this lesson.

SMTP, short for Simple Mail Transfer Protocol, is a protocol used to send emails.

So when you think of a regular contact form on a website where someone drafts a message and clicks on the submit button, that user's input is then sent, oftentimes via SMTP, to the intended recipient.

To do that, the contact form sets different headers. Those headers are interpreted by the email library on the web server, and turned into SMTP commands which are then processed by the SMTP server.

As we now know all too well, any time you have user inputs, you open up the potential for malicious injections. And in the case of SMTP, a clever attacker could attempt to inject email headers, aka SMTP headers or Mail Command Injections, which would change how the email is processed.

SMTP Header Injections impact

The results of a successful attack like this could mean that an attacker is able to:

- Send copies of emails to a 3rd party, like themselves, to steal sensitive information (like proprietary data, password reset tokens, etc)
- Spam emails
- Deliver phishing attacks that look legitimate
- Alter the content of emails
- Or even attach viruses and malware to emails being sent

Overview of SMTP

So how are SMTP Header injections possible? Let's take a closer look.

If we imagine an email as a physical letter that we send in the mail, we can imagine the email as having both an envelope and an email body.

The envelope contains this information:

```
MAIL FROM
RCPT TO
DATA
```

`MAIL FROM` is pretty straight forward – it states who the email is from.

`RCPT TO` sets the recipient of the email, and it can be used multiple times if the message is being sent to multiple people at the same time. This is important, because as we'll see in a moment, `RCPT TO` command can be manipulated in order to send out spam.

Then, `DATA` begins the email payload. The email payload has email headers and the message body separate by a single empty line.

These email headers are not part of the `SMTP` protocol, but they are interpreted by the mail client to display the email correctly, and some email handling libraries or programming languages may also interpret them.

These headers contain a `From` value and a `To` value.

The `From` value displays the sender address, although technically speaking, it could be different than the `MAIL FROM` content.

The `To` header displays the recipient, but again, it can be different than the `RCPT TO` content.

And while most libraries and languages wouldn't let you modify the contents of the envelope itself, they typically do let you modify other values that end up being translated into `SMTP` commands. For example, if you added a `BCC` header to send a "blind carbon copy" to someone else, that would translate into an `RCPT TO` command. We'll take a look at that in just a moment.

So if we look at an example of SMTP dialogue, this is what it might look like:

```
> MAIL FROM:<admin@cybr.com>
```


```
< 250 Ok
> RCPT TO:<eric@example.com>
< 250 Ok
> RCPT TO:<shawna@example.com>
< 250 Ok
> DATA
< 354 End data with <CR><LF>.<CR><LF>
> From: "Christophe Limpalair" <christophe@cybr.com>
> To: Cybr friends <cybr-friends@cybr.com>
> Date: Tue, 1 August 2020 16:02:43 -0500
> Subject: Important email to read!
>
> Hello all,
> This is a very important email used to demonstrate what SMTP looks
dialogue looks like.
> Your friend,
> Christophe
> .
< 250 Ok
```

This email, being sent by me, would be received by Eric and Shawna, and it would look like it came from my email christophe@cybr.com, instead of admin@cybr.com. They would also see the recipient as Cybr friends <cybr-friends@cybr.com>.

SMTP Header Injections

So now, let's see how we can exploit SMTP to inject our own malicious headers.

Let's say that our email code is in PHP, and this is what it looks like:

```php
<?php
$name = $_POST['name'];
$replyTo = $_POST['replyTo'];
$message = $_POST['message'];

$to = 'support@localhost';
$subject = $_POST['subject'];

$headers = "From: $name \n" .
  "Reply-To: $replyTo";
```


```
mail($to, $subject, $message, $headers);
?>
```

You're grabbing user inputs for the `name`, `replyTo`, `message`, and `subject`, before using the `mail()` function to send out an email constructed from those user inputs.

When a user goes to our contact form and submits a legitimate request, this is what it might look like:

```
POST /contact.php HTTP/1.1
Host: cybr.com

name=Christophe&replyTo=christophe@cybr.com&message=Reaching out to
discuss...&subject=Important information
```

On the other hand, an attacker could try to inject headers in order to send out spam messages, like this:

```
POST /contact.php HTTP/1.1
Host: cybr.com

name=Christophe\nbcc:
large-spam-list@example.com&replyTo=christophe@temp-mail.com&message=I feel
generous today and will match the amount of bitcoin that you send me, but
only if you are the first 20! Donate here: fake-link.com&subject=Free
Bitcoin! Hurry!
```

With this attack, we insert a newline with `\n` which should work on most UNIX and Linux systems, or `\r\n` if it's running on a Windows system, and then we add a `BCC` header that contains an email list with all of the email addresses that we wish to spam. And in our message, we send out a link to donate bitcoin, promising to match the amount donated back, because we're feeling generous.

Adding a new line, or a carriage return, is a critical step for this attack to work, and as we'll see in the defenses lesson, this can also work to our advantage to protect applications.

The email library will convert that `BCC` into `RCPT TO` commands and deliver that email to all of those emails you've injected, in addition to the intended recipient.

This is roughly what it would end up looking like:

```
> MAIL FROM:<mail-service@cybr.com>
< 250 Ok
> RCPT TO:<large-spam-list@example.com>
< 250 Ok
> RCPT TO:<christophe@temp-mail.com>
< 250 Ok
> DATA
< 354 End data with <CR><LF>.<CR><LF>
> From: "Christophe" <christophe@temp-mail.com>
> To: Support <support@cybr.com>
> Date: Tue, 1 August 2020 16:02:43 -0500
> Subject: Free Bitcoin! Hurry
>
> I feel generous today and will match the amount of bitcoin that you send
me, but only if you are the first 20! Donate here: fake-link.com
> .
< 250 Ok
```

So this is an example of an STMP Header injection being used to spam users for financial reward.

There could be far more targeted attacks, however.

For example, I've seen this attack countless times: where it looks like an email is coming from the CEO of the company, asking for "urgent funds to be transferred" or for "a gift card that they need really fast for their daughter's birthday that they almost forgot about and no one else is available to go get it!"

So while this type of attack typically doesn't cause harm to the web application or server hosting that application, it can be used to cause other types of damages. Oftentimes, attacks are used for financial gains, so that's why you can expect phishing and spam attempts.

Unfortunately, since email protocols do not have any mechanism for authentication, other types of attacks can happen with emails, so just because you're receiving spam or phishing attacks in your inbox, it doesn't necessarily mean that it was done with the techniques that we've examined, but at the same time, it could be.

IMAP injections

While we've covered SMTP injections in this lesson, there's also such a thing as IMAP injections, which we won't cover in this course, but that I wanted to provide a little bit of clarity on before wrapping up.

SMTP is used to send & deliver email, while IMAP – which stands for Internet Message Access Protocol – is used to receive email messages. IMAP is an alternative to POP, that helps sync emails between devices instead of storing emails locally on a device.

It can be possible to perform command injections on IMAP servers, so if you're curious about that I definitely recommend that you research it, but again, we won't cover it in this course at this time.

Defending against SMTP Header Injections

Slides used in this lesson.

Just like with other web-based injections, the danger of SMTP Header Injections happens when we trust user inputs.

So, let's take a look at some of the best defenses against email injections.

Use components and libraries that provide protection

In the prior chapter, I showed you an example of the PHP mail() function being vulnerable to header injections. However, there are other PHP libraries that provide protection against this type of vulnerability.

Other languages, like Python, also have components that prevent injections by default.

So this is really a matter of doing your research and choosing components or libraries that provide sufficient security, but that also doesn't mean you should blindly trust that your choice is full-proof, and you should still test for this vulnerability in your applications.

Escape user-supplied input

Because attacks require that we use newlines or carriage returns, we can look for and escape any attempts to insert those characters in user inputs. So if we see `\n` or `\r\n` in the user's input, our code should go ahead and escape that before handing it off for processing.

There are a few ways you can escape these characters, like with:

- Regex
- Libraries & components

Firewall rules

Another option that I'll mention in this chapter before wrapping up, is the ability to stop email injections via Web Application Firewalls. These firewalls can look for, and reject, requests that contain newline or carriage return characters in POST or GET requests, blocking them from even going out.

There are open-source ones, cloud-vendor specific ones, and 3rd party WAFs, available.

Conclusion and Additional Resources

Conclusion and Additional Resources

For the video version of this ebook, including more injection attacks like SQL Injections, OS Command injections, LDAP injections, and XXE / XPATH injections, check out our course on Cybr.

Be sure to follow Cybr's social media accounts for more content like this:

- LinkedIn
- Facebook
- YouTube

We also have a Discord community where you can chat in real-time with me, other course authors, and other Cybr members as well as mentors. We have forums if you'd prefer asking questions there and/or finding additional resources.

I'd also love to connect on LinkedIn, where I post regularly.

Thanks again, and hope to see you soon!

- Christophe