E-Mail Header Injection Vulnerabilities

Sai Prashanth Chandramouli, Ziming Zhao, Adam Doupé, Gail-Joon Ahn

Abstract:
E-mail Header Injection vulnerability is a class of vulnerability that can occur in web applications that use user input to construct e-mail messages. E-mail Header Injection is possible when the mailing script fails to check for the presence of e-mail headers in user input (either form fields or URL parameters). The vulnerability exists in the reference implementation of the built-in mail functionality in popular languages such as PHP, Java, Python, and Ruby. With the proper injection string, this vulnerability can be exploited to inject additional headers, modify existing headers, and alter the content of the e-mail.

ACM CCS: Security and privacy → Software and application security → Web application security

Keywords: E-mail Header Injection, Software Security

1 Introduction

The World Wide Web has single-handedly brought about a change in the way we use computers. The ubiquitous nature of the web has made it possible for anyone to access information and services anywhere and on multiple devices such as phones, laptops, personal digital assistants, TVs, and cars. This access has ushered in an era of web applications which depend on user input. While this rapid pace of development has improved the speed of dissemination of information, it does come at a cost.

Many common and well-known web application vulnerabilities, such as SQL Injection and Cross-Site Scripting [15], are command injection vulnerabilities [23], where malicious user input is used to alter the structure of a command (a SQL query in the case of SQL Injection and JavaScript code in the case of Cross-Site Scripting).

E-mail Header Injection vulnerabilities are a lesser-known command injection vulnerability. E-mail Header Injection can be considered as the e-mail equivalent of HTTP Header Injection [16]. We found that this vulnerability exists in the implementation of the built-in `mail` functionality in popular languages such as PHP, Java, Python, and Ruby. The format of e-mail messages is defined by the Simple Mail Transfer Protocol (SMTP) [19]. Each e-mail message is represented by a series of headers separated by newlines, followed by the body content (separated from the headers by two newlines). Some of these headers are mandatory (From, To, Date), but the headers can also include other information such as the Subject, CC, BCC, etc.

With the proper injection string, E-mail Header Injection vulnerabilities can be exploited by an attacker to inject additional headers, modify existing headers, or alter the contents of the e-mail—while still appearing to be from a legitimate source. E-mail Header Injection exploits allow an attacker to perform e-mail spoofing, resulting in phishing attacks *that are sent from the actual e-mail server*.

While some command injection vulnerabilities have received extensive attention from the research community, E-mail Header Injection vulnerabilities have received little focus. Therefore, we describe the causes of E-mail Header Injection vulnerabilities, the web application frameworks that are vulnerable by default, and the implications of E-mail Header Injection vulnerabilities.

2 Vulnerability Classification

E-mail Header Injection belongs to a broad class of vulnerabilities known as command injection vulnerabilities [4]. In command injection vulnerabilities, the attacker's input is able to alter the commands executed.

However, unlike its more popular command injection siblings, SQL injection (where attacker input is able to alter SQL commands) [2, 6, 20], Cross-Site Scripting (where attacker input is able to alter the HTML content of a web page) [9, 11], and HTTP Header Injection (where attacker input is able to alter the HTTP headers of a web application [10], relatively little academic

```php
$from = $_REQUEST['email'];
$subject = 'Hello XYZ';
$message = 'We need you to reset your
    password';
$to = 'xyz@example.com';

// example attack string to be
// injected as the value for
// $_REQUEST['email'] =>
// 'abc@example.com\nCC:spc@example.com'
$retValue = mail($to, $subject, $message,
    "From: $from");
// E-Mail gets sent to both
// xyz@example.com AND spc@example.com
```

Listing 1.1: PHP program E-mail Header Injection vulnerability.

```
Received: from mail.ourdomain.com
    ([62.121.130.29])
  by xyz.com (Postfix) with ESMTP id 5
     A08E52C0154
  for <abc@example.com>; Sun, 20 Mar 2016
     13:56:58 -0700 (MST)
From: abc@example.com
CC: spc@example.com
To: xyz@example.com
Subject: Hello XYZ
Date: Sun, 20 Mar 2016 13:56:58 -0700 (MST
    )

We need you to reset your password
```

Listing 1.2: SMTP headers generated by a PHP mailing script.

research is available on E-mail Header Injection vulnerabilities.

As with other vulnerabilities in this class, E-mail Header Injection is caused due to improper or nonexistent sanitization of user input. If the program that constructs e-mails from user input fails to check for the presence of e-mail headers in the user input, a malicious user—using a well-crafted payload—can control the headers set for this particular e-mail.

3 History of E-mail Header Injection

We found the first E-mail Header Injection description in a late 2004 article on phpsecure.info [26] accredited to user `tobozo@phpsecure.info` describing how an E-mail Header Injection vulnerability existed in the implementation of the `mail()` function in PHP and how it can be exploited. More recently, a blog post by Damon Kohler [12] and an accompanying wiki article [21] describe the attack vector and outline few defense measures for E-mail Header Injection vulnerabilities.

An example of vulnerable code written in PHP is shown in Listing 1.1. This code takes in user input from the PHP superglobal `$_REQUEST['email']`, and stores it in the variable `$from`, which is later passed to the `mail()` function to construct and send the e-mail.

When this code is given the malicious input `abc@example.com\nCC:spc@example.com` as the value of the `$_REQUEST['email']`, it generates the equivalent SMTP headers shown in Listing 1.2. It can be seen that the `CC` (carbon copy) header that the attacker injected appears as part of the resulting SMTP message. Thus, the SMTP server interprets this `CC` header as code rather than the intended data. This will cause an e-mail to be sent to the e-mail address specified as part of the `CC` as well.

4 Languages Affected

We investigated the e-mailing functionality of popular web application programming languages to see if they contain E-mail Header Injection vulnerabilities. This section is not intended as a complete reference of vulnerable functions and methods, but rather as a guide that specifies which parts of the language are vulnerable.

PHP was one of the first languages found to be vulnerable to E-mail Header Injection in its implementation of the `mail()` function at the time of release of PHP 4.0. According to w3techs [27], PHP is used by 81.9% of all the websites.

After 13 iterations of PHP since the 4.0 release (the current version is 7.1), the `mail()` function is yet to be fixed after 15 years. However, it is specified in the PHP documentation [17] that the `mail()` function does not protect against E-mail Header Injection. A working code sample of the vulnerability, written in PHP 5.6 (latest well-supported version), is shown in Listing 1.1.

Python A bug was filed about an E-mail Header Injection vulnerability in Python's implementation of the `email.header` library and the header parsing functions allowing newlines in early 2009, which was followed by a partial patch in early 2011.

Unfortunately, the bug fix was only for the `email.header` package, and thus exists in other frequently used packages such as `email.parser`, where both the classic `Parser()` and the newer `FeedParser()` contain E-mail Header Injection vulnerabilities even in the latest versions: `2.7.11` and `3.5`. The bug fix was also not backported to older versions of Python. There is no mention of the vulnerability in the Python documentation for either library. Contrary to PHP's behavior of overwriting existing headers, Python only recognizes the first occurrence of a header, and ignores duplicate headers. A working code sample of the vulnerability, written in Python 2.7.11, is shown in Listing 1.3.

Java has a bug report about E-mail Header Injection filed against its `JavaMail` API. A detailed write-up by Alexandre Herzog [7] contains a proof-of-concept program that exploits the API to inject headers.

Ruby From our testing, Ruby's built-in `Net::SMTP` library also has an E-mail Header Injection library. This is not documented on the library's homepage.

```python
from email.parser import Parser
import cgi
form = cgi.FieldStorage()
to = form["email"] # input() exhibits
# the same behavior
msg = """To: """ + to + """\n
From: <user@example.com>\n
Subject: Test message\n\n
Body would go here\n"""

f = FeedParser() # Parser.parsestr()
# also contains the same vulnerability
f.feed(msg)
headers = FeedParser.close(f)

# attack string =>
# 'abc@example.com\nBCC:spc@example.com'
# for form["email"]

# to:abc@example.com AND bcc:spc@example.
    com
# are added to the headers
print 'To: %s' % headers['to']
print 'BCC: %s' % headers['bcc']
```

Listing 1.3: Python program with e-mail header injection vulnerability.

```
Received: from mail.ourdomain.com
    ([62.121.130.29])
  by xyz.com (Postfix) with ESMTP id 5
      A08E52C0154
  for <abc@example.com>; Sun, 20 Mar 2016
      13:56:58 -0700 (MST)
From: abc@example.com
CC: 1@example.com, 2@example.com, 3
    @example.com
Subject: My Subject
Content-Type: multipart/mixed; boundary=
    foobar;
--foobar
Content-Type: text/html

This is the attacker's body
--foobar
To: xyz@example.com
Subject: Hello XYZ
Date: Sun, 20 Mar 2016 13:56:58 -0700 (MST
    )

We need you to reset your password
```

Listing 1.4: Exploiting E-mail Header Injection vulnerability to control recipients, subject, and body.

5 Impact of E-mail Header Injection

The impact of an E-mail Header Injection vulnerability can be far-reaching. According to w3tech, PHP, Java, Python, and Ruby (combined) account for over 85% [1] of the server-side programming languages in websites measured, and the default implementation of the e-mail functionality of these languages is vulnerable to E-mail Header Injection.

The impact of exploiting an E-mail Header Injection vulnerability depends on where in the SMTP message the attacker can control. By definition, the injection must occur in the header section of the SMTP message and not in the body. The attacker cannot alter headers that appear before the injection point, but the attacker has complete control over all the remaining SMTP message, including the message body. In this way, the attacker can completely control who receives the message (and can include multiple `CC` and `BCC` recipients), the complete body of the message, and possibly the subject of the message (if the `Subject` SMTP header is after the injection point).

The main vector for exploiting E-mail Header Injection vulnerabilities follows most command injection vulnerability exploitation: first inject the attacker's desired commands, then comment out the rest of the message. In E-mail Header Injection vulnerabilities, the attacker first includes all SMTP headers desired. These will typically be the `Subject` header to control the subject of the e-mail[2], `CC` or `BCC` headers to control the recipients of the e-mail. The `Content-type` header is used to specify that the SMTP message is a multi-part email and that the sections are separated by an attacker-specified boundary. The boundary delineates the parts of the message so that the attacker's body is the only valid part of the message, and the attacker can choose a random value for the boundary that is not present in the developer-controlled part of the SMTP message.

Thus, the attacker can completely control the e-mail. For instance, injecting the following attack payload: `abc@example.com\nCC:1@example.com, 2@example.com, 3@example.com\nSubject: My Subject\nContent-Type:multipart/mixed; boundary=foobar;\n--foobar\nContent-Type: text/html\n\nThis is the attacker's body\n--foobar` into the PHP program described in Listing 1.1 results in the SMTP message shown in Listing 1.4.

This payload is quite lengthy, so if there are limits on the size of the vulnerable variable, a simpler technique can be to use an HTML comment, if there is no HTML comments in the developer-controlled part of the message: `abc@example.com\nCC:1@example.com, 2@example.com, 3@example.com\nSubject: My Subject\nContent-Type: text/html\n\nThis is the attacker's body<!--`.

Using these techniques, an E-mail Header Injection vulnerability can be exploited to perform the following:

Phishing and Spoofing Attacks Phishing [8] (a variation of spoofing [5]) refers to an attack where the recipient of an e-mail is made to believe that the e-mail is legitimate when it was really created by a malicious party. The e-mail usually redirects the victim to a malicious

[1] A website may use more than one server-side programming language.

[2] The SMTP protocol specifies that there should only be one `Subject` header, so the attacker may not be able to alter the subject if the header is already defined. This behavior would be MUA-dependent.

website, which then steals their credentials or infects their computer with malware (via a drive-by-download).

By controlling the content of the e-mail as well as injecting arbitrary headers into an e-mail sent by a website, an attacker can leverage an E-mail Header Injection vulnerability to send phishing emails. The generated e-mail is sent from the website's mail server, therefore users (and anti-spam defenses) are more likely to trust an e-mail that is sent from the proper mail server.

Spam Networks Spam networks can use E-mail Header Injection vulnerabilities to send a large amount of e-mail from servers that are trusted. By adding additional `CC` or `BCC` headers to the generated e-mail, attackers can easily choose multiple recipients of the spam email.

Due to the e-mail being from trusted domains, recipient e-mail clients and anti-spam systems might not flag them as spam. If they do flag them as spam, then that can lead to the website being blacklisted as a spam generator (which would cause a Denial of Service on the vulnerable web application). Unlike traditional botnets sending spam, which typically do so through residential ISPs, E-mail Header Injection vulnerabilities exist on web applications on business networks, which have more bandwidth than residential networks. Thus, we believe that attackers could exploit E-mail Header Injection vulnerabilities to generate substantial amounts of spam.

Information Extraction E-mails can contain sensitive data that is meant to be accessed only by the user. Due to an E-mail Header Injection vulnerability, an attacker can add a `BCC` header, and the e-mail server will send a copy of the private e-mail to the attacker, thereby extracting important information. User privacy can thus be compromised, and loss of private information can lead to other attacks.

Denial of Service Denial of service attacks (DoS), can be caused by exploiting an E-mail Header Injection vulnerability. The ability to send many e-mails by injecting one header field can result in overloading the mail server and cause crashes or instability.

E-mail Header Injection vulnerabilities are particularly impactful because they exploit the trust that mail clients have in the vulnerable mail server. Attackers can take advantage of this trust to facilitate their malicious behaviors. In addition, active exploitation of E-mail Header Injection vulnerabilities degrades this trust, which can be difficult to recover.

6 Related Work

Our work on studying E-mail Header Injection vulnerabilities is related to other injection based attacks, such as SQL Injection [2, 6, 20], Cross-Site Scripting [9, 11], HTTP Header Injection [10], and the related Simple Mail Transfer Protocol (SMTP) Injection [25].

The attack described by Terada [25] is one that attacks the underlying SMTP mail servers by injecting SMTP commands (which are closely related to E-Mail Headers and usually have a one-to-one mapping, e.g., `To` e-mail header has a corresponding `To` SMTP header) to exploit the SMTP server's pipelining mechanism. Terada also describes proof-of-concept attacks against certain mailing libraries such as `Ruby Mail` and `JavaMail`. This attack, although trying to achieve a similar result, is distinctly different from ours.

The first documented article on E-mail Header Injection vulnerabilities is a late 2004 article on phpsecure.info [26] accredited to user `tobozo@phpsecure.info` describing how this vulnerability existed in the reference implementation of the mail function in PHP, and how it can be exploited. Following this, we found other blog posts [3, 12, 13, 14, 18], each describing how to exploit the vulnerability by using newlines to camouflage headers inside user input. However, none of these articles have categorized and described both the cause of the vulnerability as well as the impact.

Another blog post written by user Voxel@Night on Vexatious Tendencies [24], recounts an actual exploit against a WordPress plugin, `Contact Form`, with a proof of concept[3]. It also showcases the vulnerable code in the plugin that causes this vulnerability to be present. This article targets one plugin. Neither does it inform the creators of the plugin to fix the discovered vulnerability. E-mail Header Injection was described briefly by Stuttard and Pinto in their book, "*The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*" [22]. The book, however, does not go into detail on the attack.

7 Conclusion

In this chapter, we have discussed a vulnerability class that received little attention from the research community—E-mail Header Injection vulnerabilities. After discussing the causes and implications of these vulnerabilities, we hope that this work will serve to motive the research community to study and address this vulnerability class. Because E-mail Header Injection vulnerabilities can be used to facility spam and phishing attacks, these vulnerabilities can potentially affect all Internet users.

[3] Note that this plugin is used actively on 300,000 websites (according to [1]), but is yet to be fixed.

Literaturverzeichnis

[1] BestWebSoft. Contact Form by BestWebSoft WordPress Plugins. https://wordpress.org/plugins/contact-form-plugin/, 2016.

[2] S. W. Boyd and A. D. Keromytis. Sqlrand: Preventing sql injection attacks. In *Applied Cryptography and Network Security*, pages 292–302. Springer, 2004.

[3] B. Calin. Email Header Injection Web Vulnerability - Acunetix. https://www.acunetix.com/blog/articles/email-header-injection-web-vulnerability-detection/, 2013.

[4] Common Weakness Enumeration. CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection'). https://cwe.mitre.org/data/definitions/77.html.

[5] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An internet con game. *Software World*, 28(2):6–8, 1997.

[6] W. G. Halfond, J. Viegas, and A. Orso. A classification of sql-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, volume 1, pages 13–15. IEEE, 2006.

[7] A. Herzog. Full Disclosure: JavaMail SMTP Header Injection via method setSubject [CSNC-2014-001], 2014.

[8] M. Jakobsson and S. Myers. *Phishing and countermeasures: understanding the increasing problem of electronic identity theft*. John Wiley & Sons, 2006.

[9] T. Jim, N. Swamy, and M. Hicks. Defeating script injection attacks with browser-enforced embedded policies. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 601–610, New York, NY, USA, 2007. ACM.

[10] M. Johns and J. Winter. Requestrodeo: Client side protection against session riding. In *Proceedings of the OWASP Europe 2006 Conference*, 2006.

[11] A. Klein. [DOM Based Cross Site Scripting or XSS of the Third Kind] Web Security Articles - Web Application Security Consortium, 2005.

[12] D. Kohler. damonkohler: Email Injection. http://www.damonkohler.com/2008/12/email-injection.html, 2008.

[13] A. Mohamed. PHP Email Injection Example - InfoSec Resources. http://resources.infosecinstitute.com/email-injection/, 2013.

[14] J. Nicol. Securing PHP Contact Forms. http://jonathannicol.com/blog/2006/12/09/securing-php-contact-forms/, 2006.

[15] OWASP. OWASP Top Ten Project. https://www.owasp.org/index.php/OWASP_Top_10, 2013.

[16] OWASP. OWASP - HTTP header injection. https://www.owasp.org/index.php/HTTP_Response_Splitting, 2016.

[17] PHP-Manual. PHP mail - Send mail. http://php.net/manual/en/function.mail.php, 2016.

[18] A. Pope. Prevent Contact Form Spam Email Header Injection — Storm Consultancy Web Design Bath, 2008.

[19] P. W. Resnick. Internet Message Format - RFC 5322. 2008.

[20] A. Sadeghian, M. Zamani, and A. A. Manaf. A taxonomy of sql injection detection and prevention techniques. In *Informatics and Creative Multimedia (ICICM), 2013 International Conference on*, pages 53–56. IEEE, 2013.

[21] Email Injection - Secure PHP Wiki. http://securephpwiki.com/index.php/EmailInjection, 2010.

[22] D. Stuttard and M. Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. John Wiley & Sons, 2011.

[23] Z. Su and G. Wassermann. The essence of command injection attacks in web applications. In *ACM SIGPLAN Notices*, volume 41, pages 372–382. ACM, 2006.

[24] V. Tendencies. WordPress Plugin Vulnerability Dump Part 2 — Vexatious Tendencies. https://vexatioustendencies.com/wordpress-plugin-vulnerability-dump-part-2/, 2014.

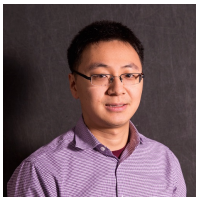
[25] T. Terada. SMTP Injection via recipient email addresses. *MBSD White Paper*, December 2015.

[26] Tobozo. Mail headers injections with PHP. http://www.phpsecure.info/v2/article/MailHeadersInject.en.php, 2004.

[27] W3techs. Usage Statistics and Market Share of PHP for Websites, February 2016. http://w3techs.com/technologies/details/pl-php/all/all, 2016.

Sai Prashanth Chandramouli has a Masters in Computer Science from Arizona State University, with a thesis on E-mail Header Injection vulnerability, which he developed under the guidance of Dr. Adam Doupé. His interests include web security and computational creativity.

Address: Arizona State University, E-Mail:saipc@asu.edu



Dr. Ziming Zhao is an assistant research professor in the School of Computing, Informatics, and Decision Systems Engineering, Ira A. Fulton Schools of Engineering, Arizona State University. His research interests include system and network security and cybercrime analysis. Dr. Zhao received a Ph.D in Computer Science from Arizona State University (ASU). He is a member of IEEE and the ACM.

Address: Arizona State University, E-Mail:zzhao30@asu.edu

Dr. Adam Doupé is an Assistant Professor in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. His research interests include vulnerability analysis, web security, mobile security, and hacking competitions, which has been supported by the National Science Foundation.

Address: Arizona State University, E-Mail:zzhao30@asu.edu

Dr. Gail-Joon Ahn is currently a professor of computer science and engineering in the School of Computing, Informatics, and Decision Systems Engineering and the director of Center for Cybersecurity and Digital Forensics, Arizona State University. His research interests include information and systems security, vulnerability and risk management, access control, and security architecture for distributed systems, which has been supported by National Science Foundation, Department of Defense, Office of Naval Research, Army Research Office, Department of Justice, and private sectors including Allstate, Bank of America, Hewlett Packard, Microsoft, Robert Wood Johnson Foundation, Cisco, GoDaddy, and Intel. He received the Department of Energy Early Career Investigator Award and the Educator of the Year Award given by the Federal Information Systems Security Educators Association in 2005.

Address: Arizona State University, E-Mail:gahn@asu.edu