

OX App Suite / OX Documents 7.10.x XSS / SSRF

Authored by [Martin Heiland, notoriousrip, Stuart Redman](#)

Posted Jan 8, 2021

OX App Suite and OX Documents suffer from server-side request forgery and multiple cross site scripting vulnerabilities. Various versions are affected including 7.10.4 and 7.10.3.

tags | [exploit](#), [vulnerability](#), [xss](#)

advisories | [CVE-2020-24700](#), [CVE-2020-24701](#)

SHA-256 | [ba8c16584bc43d579279e941f2d796ec74153f6debe5a7df85b435f86196a43c](#)

[Download](#) | [Favorite](#) | [View](#)

Related Files

Share This

Like

Twitter

LinkedIn

Reddit

Digg

StumbleUpon

[Change Mirror](#)[Download](#)

Product: OX App Suite / OX Documents
Vendor: OX Software GmbH

Internal reference: MWB-423
Vulnerability type: Server-Side Request Forgery (CWE-918)
Vulnerable version: 7.10.4 and earlier
Vulnerable component: backend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.6.3-rev55, 7.10.4-rev9
Vendor notification: 2020-06-26
Solution date: 2020-09-23
Public disclosure: 2021-01-07
Researcher Credits: Stuart Redman
CVE reference: CVE-2020-24700
CVSS: 6.4 (CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N)

Vulnerability Details:
The OAuth Proxy capability, used to exchange data with third-party services such as Twitter, can be abused to craft requests to services which are prohibited. These services may reside within a protected network and could be exposed using this technique. The code to check for allowed domains did not account for certain URL constructs.

Risk:
Malicious users can trigger network requests to web services outside of the expected trust boundary, for example services within a restricted network to which the OX App Suite middleware node has access. In case such services do not have further access control, a malicious user could retrieve web service content from them. The vulnerability allows to control request type and headers sent to those services.

Steps to reproduce:
1. Connect your OX App Suite account to an OAuth-enabled service like Twitter
2. Forge API requests via /api/oauth/proxy containing payload related to internal services
3. API response will contain an error but also the retrieved content for the internal service

Proof of concept:
PUT https://example.com/appsuite/api/oauth/proxy?api=com.openexchange.oauth.twitter&session=XYZ
{"url":"https://twitter.com@internal.example.com","params":{"count":10,"include_entities":true}}

Solution:
We have improved detection of user-provided payload when checking against access lists. Regardless of this fix we suggest tight network segmentation, egress traffic filtering and access controls for any kind of service.

Internal reference: MWB-460
Vulnerability type: Server-Side Request Forgery (CWE-918)
Vulnerable version: 7.10.3 and earlier
Vulnerable component: backend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.6.3-rev55, 7.10.4-rev9
Vendor notification: 2020-07-07
Solution date: 2020-09-23
Public disclosure: 2021-01-07
CVE reference: CVE-2020-24700
CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N)

Vulnerability Details:
External mail account discovery allows malicious users to append arbitrary URL paths to mail addresses. In combination with malicious auto-configuration DNS records, this can be abused to access web services outside of the expected trust boundary, regardless of existing blocklists.

Risk:
Malicious users can trigger network requests to web services outside of the expected trust boundary, regardless of existing blocklists. This may be used to probe for services and paths within a restricted network to which the OX App Suite middleware node has access and potentially ease further attacks.

Steps to reproduce:
1. Setup a DNS A record for autoconfig.example.com, pointing to a local addresses like 127.0.0.1
2. Use the "external mail account" feature to setup a mail account for this domain
3. Append URL paths to the mail address, e.g. foo@example.com/ssrf/ping

Proof of concept:
DNS lookup will return "127.0.0.1" and OX App Suite will append the URL fragment of the mail address, resulting in a GET request to http://127.0.0.1/ssrf/ping?emailaddress=foo@example.com.

Solution:
We restricted the ability to access blocked networks when performing autoconfig lookups.

Internal reference: MWB-492
Vulnerability type: Cross-Site Scripting (CWE-80)
Vulnerable version: 7.10.3 and earlier
Vulnerable component: backend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.6.3-rev55, 7.10.4-rev9
Vendor notification: 2020-07-20
Solution date: 2020-09-23
Public disclosure: 2021-01-07
CVE reference: CVE-2020-24701
CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N)

Vulnerability Details:
The "debug" option for the /apps/manifests endpoint included request parameters in its response, without using HTML escaping.

Risk:
Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to follow a hyperlink.

Steps to reproduce:
1. Create a link to the /apps/manifest endpoint using the debug option and append malicious script code
2. Make a user open this link, for example through social engineering

Proof of concept:
https://example.com/ajax/apps/manifests?action=all&format=debug&xss=%3Cscript%3Ealert(%22XSS%22);%3C/script%3E

Solution:
We now escape any user-provided content when creating the debug response.

Search ...

Follow us on Twitter

Subscribe to an RSS Feed

File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr
Sa					
				1	2
3					
4	5	6	7	8	9
10					
11	12	13	14	15	16
17					
18	19	20	21	22	23
24					
25	26	27	28	29	30
31					

Top Authors In Last 30 Days

Red Hat 150 files

Ubuntu 68 files

LiquidWorm 23 files

Debian 16 files

malvuln 11 files

nu11security 11 files

Gentoo 9 files

Google Security Research 6 files

Julien Ahrens 4 files

T. Weber 4 files

File Tags

ActiveX (932)	December 2022
Advisory (79,754)	November 2022
Arbitrary (15,694)	October 2022
BBS (2,859)	September 2022
Bypass (1,619)	August 2022
CGI (1,018)	July 2022
Code Execution (6,926)	June 2022
Conference (673)	May 2022
Cracker (840)	April 2022
CSRF (3,290)	March 2022
DoS (22,602)	February 2022
Encryption (2,349)	January 2022
Exploit (50,359)	Older

File Inclusion (4,165)

File Upload (946)

Firewall (821)

Info Disclosure (2,660)

Intrusion Detection (867)

Java (2,899)

JavaScript (821)

Kernel (6,291)

Local (14,201)

Magazine (586)

Overflow (12,419)

Perl (1,418)

PHP (5,093)

Proof of Concept (2,291)

Protocol (3,435)

Python (1,467)

Remote (30,044)

Root (3,504)

Ruby (594)

Scanner (1,631)

Security Tool (7,777)

Shell (3,103)

Shellcode (1,204)

Sniffer (886)

File Archives

December 2022

November 2022

October 2022

September 2022

August 2022

July 2022

June 2022

May 2022

April 2022

March 2022

February 2022

January 2022

Older

Systems

AIX (426)

Apple (1,926)

BSD (370)

CentOS (55)

Cisco (1,917)

Debian (6,634)

Fedora (1,600)

FreeBSD (1,242)

Gentoo (4,272)

HPUX (878)

IOS (330)

iPhone (108)

IRIX (220)

Juniper (67)

Linux (44,315)

Mac OS X (684)

Mandriva (3,105)

NetBSD (255)

OpenBSD (479)

RedHat (12,469)

Slackware (941)

Solaris (1,607)

<div>--- Internal reference: MWB-493 Vulnerability type: Cross-Site Scripting (CWE-80) Vulnerable version: 7.10.4 and earlier Vulnerable component: backend Report confidence: Confirmed Solution status: Fixed by Vendor Fixed version: 7.6.3-rev55, 7.10.4-rev9 Vendor notification: 2020-07-20 Solution date: 2020-09-23 Public disclosure: 2021-01-07 CVE reference: CVE-2020-24701 CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N) Vulnerability Details: The logic for determining safe content could be bypassed by providing unknown values for content-disposition while requesting a shared file. In case the file contained malicious script code, this would be executed. Risk: Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to follow a hyperlink. Steps to reproduce: 1. Create a HTML file with malicious JS code and upload it to Drive 2. Create a public sharing link 3. Modify this link to contain a unexpected content_disposition parameter value 4. Make the victim follow this link Proof of concept: https://example.com/ajax/share/<share-token>?delivery=viewscontent_disposition=foo Solution: We improved the detection mechanism to neglect user-specified parameter values. ---</div>	
<div>Internal reference: MWB-494 Vulnerability type: Cross-Site Scripting (CWE-80) Vulnerable version: 7.10.4 and earlier Vulnerable component: backend Report confidence: Confirmed Solution status: Fixed by Vendor Fixed version: 7.6.3-rev55, 7.10.4-rev9 Vendor notification: 2020-07-21 Solution date: 2020-09-23 Public disclosure: 2021-01-07 CVE reference: CVE-2020-24701 CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N) Vulnerability Details: Risk: Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to follow a hyperlink. Steps to reproduce: 1. Include malicious script code within external content like a vcard file 2. Attach this file to a mail and use the conversion API to create a managed distributed file 3. Find out the UUID reference to this managed "distributedFile" 4. Make the victim open this direct reference as hyperlink Solution: We now require user-specific authentication to access this API endpoint and request managed distributed files. ---</div>	
<div>Internal reference: MWB-520 Vulnerability type: Cross-Site Scripting (CWE-80) Vulnerable version: 7.10.4 and earlier Vulnerable component: backend Report confidence: Confirmed Solution status: Fixed by Vendor Fixed version: 7.6.3-rev55, 7.10.4-rev9 Vendor notification: 2020-07-30 Solution date: 2020-09-23 Public disclosure: 2021-01-07 CVE reference: CVE-2020-24701 CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N) Vulnerability Details: Binary files could be requested for "inline" delivery, which results in content processing within the browser. This allows to inject and execute script code within a "binary" file. Risk: Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to follow a hyperlink. Steps to reproduce: 1. Craft a malicious HTML/JS file, upload it to Drive as binary content (application/octet-stream or similar) 2. Create a sharing link and modify its "delivery" parameters 3. Make a victim follow this link Solution: We removed the undocumented parameter from requests to shared content. ---</div>	
<div>Internal reference: MWB-583 Vulnerability type: Cross-Site Scripting (CWE-80) Vulnerable version: 7.10.4 and earlier Vulnerable component: backend Report confidence: Confirmed Solution status: Fixed by Vendor Fixed version: 7.6.3-rev55, 7.10.3-rev22, 7.10.4-rev9 Vendor notification: 2020-08-31 Solution date: 2020-09-23 Public disclosure: 2021-01-07 CVE reference: CVE-2020-24701 CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N) Vulnerability Details: File names of inline images are not escaped or sanitized when creating a raw representation of the mail content, for example when displaying huge HTML mails through "Show entire message" Risk: Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to follow a hyperlink. Steps to reproduce: 1. Craft a large HTML E-Mail and include inline images 2. Chose a inline image filename that contains urlencoded script code 3. Send the mail to the victim and make it click "Show entire message" Solution: We now HTML-escape inline image file names when adding them as HTML attribute. ---</div>	
<div>Internal reference: OXUIB-400 Vulnerability type: Cross-Site Scripting (CWE-80) Vulnerable version: 7.10.4 and earlier Vulnerable component: frontend Report confidence: Confirmed Solution status: Fixed by Vendor Fixed version: 7.10.4-rev8 Vendor notification: 2020-08-13</div>	

Spoof (2,166)	SUSE (1,444)
SQL Injection (16,102)	Ubuntu (8,199)
TCP (2,379)	UNIX (9,159)
Trojan (686)	UnixWare (185)
UDP (876)	Windows (6,511)
Virus (662)	Other
Vulnerability (31,136)	
Web (9,365)	
Whitepaper (3,729)	
x86 (946)	
XSS (17,494)	
Other	

Solution date: 2020-09-23
Public disclosure: 2021-01-07
CVE reference: CVE-2020-24701
CVSS: 5.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N)

Vulnerability Details:
Error responses for loading frontend apps were not properly escaped, which allows malicious users to inject script code.

Risk:
Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to follow a hyperlink.

Steps to reproduce:
1. Construct a URL pointing to the app loading mechanism
2. Include malicious JS code within this link
3. Make a victim open the link

Proof of concept:
`https://example.com/appsuite/#!/$app=io.ox/files:foo,xx/../../xxx");alert("XSS");//`

Solution:
We now escape error responses when loading an app fails.

Internal reference: OXUIB-401
Vulnerability type: Cross-Site Scripting (CWE-80)
Vulnerable version: 7.10.4 and earlier
Vulnerable component: frontend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.10.4-rev8
Vendor notification: 2020-08-07
Solution date: 2020-09-23
Public disclosure: 2021-01-07
Researcher Credits: notoriousrip
CVE reference: CVE-2020-24701
CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N)

Vulnerability Details:
A special "mail://" URL handler is used to allow back-links from tasks to E-Mail, in case a task was created as "reminder" for a mail. Contents of this URL would not be properly sanitized and added to the "notes" section of tasks. Attackers in a position to create such malicious tasks could place script code within that URL.

Risk:
Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to interact with malicious tasks either shared within the same context or manually imported.

Steps to reproduce:
1. Craft a malicious task, containing script code as "Note"
2. Make the user import the task or share it within the same context
3. Make the user interact with the tasks note

Proof of concept:
`mail://hello"onmouseover=alert(document.cookie)\;"@example.com`

Solution:
We now sanitize task notes before adding them to DOM.

Internal reference: OXUIB-411
Vulnerability type: Cross-Site Scripting (CWE-80)
Vulnerable version: 7.10.4 and earlier
Vulnerable component: frontend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.10.3-rev20, 7.10.4-rev11
Vendor notification: 2020-08-18
Solution date: 2020-09-23
Public disclosure: 2021-01-07
CVE reference: CVE-2020-24701
CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N)

Vulnerability Details:
Data like names of contacts could contain script code. When using the mobile mode (e.g. on a smartphone) and searching for contacts, this script code would be appended to DOM in an unsafe way without sanitizing its content.

Risk:
Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to interact with malicious content either shared within the same context or manually imported.

Steps to reproduce:
1. Create or import a contact with script-code as name
2. In mobile mode, use search and look up that contact

Solution:
We use DOMPurify to clean up values before using them as search results in mobile mode.

Internal reference: OXUIB-412
Vulnerability type: Cross-Site Scripting (CWE-80)
Vulnerable version: 7.10.4 and earlier
Vulnerable component: frontend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.10.3-rev19, 7.10.4-rev8
Vendor notification: 2020-08-18
Solution date: 2020-09-23
Public disclosure: 2021-01-07
CVE reference: CVE-2020-24701
CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N)

Vulnerability Details:
Data like the location of appointments could contain script code. When using the mobile mode (e.g. on a smartphone) and searching for appointments, this script code would be appended to DOM in an unsafe way without sanitizing its content.

Risk:
Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to interact with malicious content either shared within the same context or manually imported.

Steps to reproduce:
1. Create or import a appointment with script-code as location
2. In mobile mode, use search and look up that appointment

Solution:
We use DOMPurify to clean up values before using them as search results in mobile mode.

Internal reference: OXUIB-421
Vulnerability type: Cross-Site Scripting (CWE-80)
Vulnerable version: 7.10.4 and earlier
Vulnerable component: frontend
Report confidence: Confirmed
Solution status: Fixed by Vendor
Fixed version: 7.10.3-rev19, 7.10.4-rev8
Vendor notification: 2020-08-20
Solution date: 2020-09-23
Public disclosure: 2021-01-07
CVE reference: CVE-2020-24701
CVSS: 4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N)

Vulnerability Details:
Data like subjects of tasks could contain script code. When using the mobile mode (e.g. on a smartphone) and searching for tasks, this script code would be appended to DOM in an unsafe way without sanitizing its content.

Risk:
Malicious script code can be executed within a users context. This can lead to session hijacking or triggering unwanted actions via the web interface (e.g. redirecting to a third-party site). To exploit this an attacker would require the victim to interact with malicious content either shared within the same context or manually imported.

Steps to reproduce:
1. Create or import a task with script-code as subject
2. In mobile mode, use search and look up that task

Solution:
We use DOMPurify to clean up values before using them as search results in mobile mode.

[Login](#) or [Register](#) to add favorites

packet storm
© 2022 Packet Storm. All rights reserved.

Site Links

[News by Month](#)

[News Tags](#)

[Files by Month](#)

[File Tags](#)

[File Directory](#)

About Us

[History & Purpose](#)

[Contact Information](#)


[Terms of Service](#)


[Privacy Statement](#)

[Copyright Information](#)

Hosting By

[Rokasec](#)

 [Follow us on Twitter](#)

 [Subscribe to an RSS Feed](#)